本资源来自数缘社区

http://maths.utime.cn:81

欢迎来到数缘社区。本社区是一个高等数学及密码学的技术性论坛，由山东大学数学院研究生创办。在这里您可以尽情的遨游数学的海洋。作为站长，我诚挚的邀请您加入，希望大家能一起支持发展我们的论坛，充实每个版块。把您宝贵的资料与大家一起分享！

数学电子书库

每天都有来源于各类网站的与数学相关的新内容供大家浏览和下载，您既可以点击左键弹出网页在线阅读，又可以点右键选择下载。现在书库中藏书 1000 余本。如果本站没有您急需的电子书，可以发帖说明，我们有专人负责为您寻找您需要的电子书。

密码学论文库

国内首创信息安全专业的密码学论文库，主要收集欧密会（Eurocrypt）、美密会（Crypto）、亚密会（Asiacrypt）等国内外知名论文。现在论文库中收藏论文 4000 余篇（包括论文库版块 700 余篇、论坛顶部菜单“密码学会议论文集”3000 余篇）。如果本站没有您急需的密码学论文，可以发帖说明，我们有专人负责为您寻找您需要的论文。

提示：本站已经收集到 1981－2003 年欧密会、美密会全部论文以及 1997 年－2003 年五大会议全部论文（欧密会、美密会、亚密会、PKC、FSE）。

数学综合讨论区

论坛管理团队及部分会员来源于山东大学数学院七大专业（基础数学、应用数学、运筹学、控制论、计算数学、统计学、信息安全），在数学方面均为思维活跃、成绩优秀的研究生，相信会给您的数学学习带来很大的帮助。

密码学与网络安全

山东大学数学院的信息安全专业师资雄厚，前景广阔，具有密码理论、密码技术与网络安全技术三个研究方向。有一大批博士、硕士及本科生活跃于本论坛。本版块适合从事密码学或网络安全方面学习研究的朋友访问。

网络公式编辑器

数缘社区公式编辑器采用 Latex 语言，适用于任何支持图片格式的论坛或网页。在本论坛编辑好公式后，您可以将自动生成的公式图片的链接直接复制到你要发的帖子里以图片的形式发表。

如果您觉得本站对您的学习和成长有所帮助，请把它添加到您的收藏夹。如果您对本论坛有任何的意见或者建议，请来论坛留下您宝贵的意见。

附录 A：本站电子书库藏书目录
http://maths.utime.cn:81/bbs/dispbbs.asp?boardID=18&ID=2285

附录 B：版权问题
数缘社区所有电子资源均来自网络，版权归原作者所有，本站不承担任何版权责任。

国外著名高等院校
信息科学与技术优秀教材

密码学基础

FOUNDATIONS OF CRYPTOGRAPHY

Basic Tools

〔以色列〕Oded Goldreich 著
温巧燕 杨义先 译
罗守山 张振涛 审校



中文版

人民邮电出版社
POSTS & TELECOM PRESS

封面设计：胡平利

国外著名高等院校信息科学与技术优秀教材

密码学基础

FOUNDATIONS OF CRYPTOGRAPHY *Basic Tools*

中文版



密码学涉及解决安全问题的计算系统的概念化、定义以及构造。密码系统的设计必须基于坚实的基础。本书对这一基础问题给出了系统而严格的论述，用已有工具来定义密码系统的目标并解决新的密码问题。本书集中讨论基本数学问题：计算复杂性（单向函数）、伪随机数以及零知识证明。本书的重点在于澄清基本概念并论述解决密码问题的可行性，而不侧重于描述某种具体方法。

本书可作为密码学的教材和专业人员的参考书。本书假定读者熟悉算法设计与算法分析；一些复杂性理论和概率的知识也是有用的。

Oded Goldreich 是 Weizmann 学院计算机科学教授，也是 Meyer W. Weisgal Professorial Chair 的成员。作为一名活跃的学者，他已经发表了大量密码学论文，是密码学领域公认的世界级专家。他还是 *Journal of Cryptology*、*SIAM Journal on Computing* 杂志的编辑，出版了《现代密码学、概率证明与伪随机数》一书。

TM

ISBN 7-115-10355-0

9 787115 103550 >

人民邮电出版社网址 www.ptpress.com.cn

ISBN7-115-10355-0/TP·2914
定价：35.00 元

国外著名高等院校信息科学与技术优秀教材

密码学基础

[以色列] Oded Goldreich　著

温巧燕　杨义先　译

罗守山　张振涛　审校

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

密码学基础/（以）戈德里克（Goldreich，O.）编著；温巧燕，杨义先译.—北京：人民邮电出版社，2003.9
国外著名高等院校信息科学与技术优秀教材
ISBN 7－115－10355－0

Ⅰ. 密… Ⅱ. ①戈… ②温… ③杨… Ⅲ. 密码—理论—高等学校—教材 Ⅳ. TN918.1

中国版本图书馆 CIP 数据核字（2003）第 056367 号

版 权 声 明

Oded Goldreich: Foundations of Cryptography Basic Tools

Copyright © Oded Goldreich 2001

Authorized translation from the English language edition published by the Press Syndicate of the University of Cambridge.

All rights reserved.

本书中文简体字版由英国剑桥大学出版社授权人民邮电出版社出版。未经出版者书面许可，对本书任何部分不得以任何方式复制或抄袭。

版权所有，侵权必究。

国外著名高等院校信息科学与技术优秀教材

密码学基础

◆ 著　[以色列] Oded Goldreich
译　温巧燕　杨义先
审　校　罗守山　张振涛
责任编辑　李　际

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线　010-67132705
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：18.5
字数：446 千字　2003 年 9 月第 1 版
印数：1-4 000 册　2003 年 9 月北京第 1 次印刷

著作权合同登记　图字：01 - 2002 - 0398 号

ISBN 7-115-10355-0/TP・2914

定价：35.00 元

本书如有印装质量问题，请与本社联系　电话：(010) 67129223

内容提要

密码学涉及解决安全问题的计算系统的概念化、定义以及构造。密码系统的设计必须基于坚实的基础。本书对这一基础问题给出了系统而严格的论述：用已有工具来定义密码系统的目标并解决新的密码问题。本书集中讨论：计算复杂性（单向函数）、伪随机数以及零知识证明。本书的重点在于澄清基本概念并论述解决密码问题的可行性，而不侧重于描述某种具体方法。

本书可作为密码学、应用数学、信息安全等专业的教材，也可作为相关专业人员的参考用书。

出版说明

2001年，教育部印发了《关于“十五”期间普通高等教育教材建设与改革的意见》。该文件明确指出，“九五”期间原国家教委在“抓好重点教材，全面提高质量”方针指导下，调动了各方面的积极性，产生了一大批具有改革特色的新教材。然而随着科学技术的飞速发展，目前高校教材建设工作仍滞后于教学改革的实践，一些教材内容陈旧，不能满足按新的专业目录修订的教学计划和课程设置的需要。为此该文件明确强调，要加强国外教材的引进工作。当前，引进的重点是信息科学与技术和生物科学与技术两大学科的教材。要根据专业（课程）建设的需要，通过深入调查、专家论证，引进国外优秀教材。要注意引进教材的系统配套，加强对引进教材的宣传，促进引进教材的使用和推广。

邓小平同志早在1977年就明确指出：“要引进外国教材，吸收外国教材中有益的东西。”随着我国加入WTO，信息产业的国际竞争将日趋激烈，我们必须尽快培养出大批具有国际竞争能力的高水平信息技术人才。教材是一个很关键的问题，国外的一些优秀教材不但内容新，而且还提供了很多新的研究方法和思考方式。引进国外原版教材，可以促进我国教学水平的提高，提高学生的英语水平和学习能力，保证我们培养出的学生具有国际水准。

为了贯彻中央“科教兴国”的方针，配合国内高等教育教材建设的需要，人民邮电出版社约请有关专家反复论证，与国外知名的教材出版公司合作，陆续引进一些信息科学与技术优秀教材。第一批教材针对计算机专业的主干核心课程，是国外著名高等院校所采用的教材，教材的作者都是在相关领域享有盛名的专家教授。这些教材内容新，反映了计算机科学技术的最新发展，对全面提高我国信息科学与技术的教学水平必将起到巨大的推动作用。

出版国外著名高等院校信息科学与技术优秀教材的工作将是一个长期的、坚持不懈的过程，我社网站（www.ptpress.com.cn）上介绍了我们陆续推出的图书的详细情况，敬请关注。希望广大教师和学生将使用中的意见和建议及时反馈给我们，我们将根据您的反馈不断改进我们的工作，推出更多更好的引进版信息科学与技术教材。

人民邮电出版社

译者序

人类对密码的研究与应用已有几千年的历史，但密码学作为一门系统科学则仅仅是上个世纪 50 年代的事情。1949 年，信息论的创始者 Shannon 发表的著名论文“保密通信的信息理论”将密码学的研究纳入了科学的轨道，使密码学正式成为一门科学。1976 年，美国著名学者 Diffie 和 Hellman 的经典论文“密码学的新方向”奠定了公钥密码学的基础，它标志着密码学的研究和实践由传统走向现代。1977 年，美国公布并实施的数据加密标准 DES，揭开了密码学的神秘面纱，使密码学的研究和应用从秘密走向公开，吸引了众多学者和技术人员对其进行研究，从此密码学成为了一门蓬勃发展的学科。而今，随着互联网的迅速普及，信息安全问题引起了全人类的重视，密码学作为信息安全的核心更得到了长足发展。现在，大多数国家或地区都已经成立了密码学会，国际国内各种密码学术会议频繁召开，许多大学都开设了密码学专业并进行本科生和研究生的招生和培养。总之，与密码学相关的众多活动极大地促进了它的研究与应用。如今，密码学理论研究已比较成熟，国内外已出版了大量有关密码学的书籍，许多著名学者都基于自己的研究出版了专著，这些著作各具特色，其中有不少佳作适合做教材。Oded Goldreich 教授所著的《密码学基础》一书就是一本很有特色的密码学教科书，其最显著的特点就是“基础”二字，正如该书前言中引用的那样：

“没有根基也许可以造一座小屋，但绝不能造一座坚固的大厦。”

密码学所关心的是构造一些能抵抗任何攻击的机制，这种密码机制的设计是一项十分困难的任务，它必须建立在坚实的基础之上。本书对这一基础问题给出了系统而严格的论述。

根深才能叶茂。相信这本书对于希望在密码学的殿堂里有所造诣的学子们大有裨益。这正是我们花费大量精力翻译本书的初衷。

参加本书翻译工作的还有郭奋卓、张洁、麻小宁、安宁、付华等，特别是郭奋卓博士协助我们完成了全书的统稿和审校。本书在翻译过程中，还得到了北京邮电大学信息安全中心钮心忻博士、徐国爱博士等的支持与协助，在此一并对他们表示衷心地感谢。

本书的出版得到了国家973项目（编号：G1999035804）、国家自然科学基金重大项目（编号：90204017）和国家863项目（编号：2002AA143041）的资助，在此表示感谢。

杨义先
北京邮电大学
2003年于北京

前 言

没有根基也许可以造一座小屋，
但绝不能造一座坚固的大厦。
Eng. Isidor Goldreich（1906～1995）

密码学所关心的是构造一些能抵抗任何攻击的机制。构造这样的机制是为了使系统在遭受企图让它们偏离规定功能的恶意攻击下，依旧能保持期望的功能。

密码机制的设计是一个十分困难的任务。人们不能仅依靠有关系统运行环境的典型状态的直观知识。的确，攻击系统的*对手*（adversary）可能企图控制操作环境使之成为非典型状态。人们不可能满足于设计用来抵抗特定攻击的反攻击手段，因为对手（将在系统设计完成后才行动）可能试图以和设计者所预料的大不相同的方式来攻击此机制。虽然上述断言似乎是不证自明的，但是，仍有一些人期望在实践中忽略这些赘述不会导致实际损失。事实证明这样的愿望很少能达到；基于伪装的密码机制迟早会被破坏。

按照上述观点，我们认为就对手可能使用的特定*策略*（strategy）进行假设是没有什么意义的，惟一合理的假设是对手的计算*能力*（ability）。进一步说来，我们认为密码系统的设计必须建立在*坚实的基础*（firm foundation）之上，专门方法和试探方法则是非常危险的。当设计者对机制即将运行的环境十分了解时，试探方法可能有意义，但是密码机制将不得不在恶意选择的环境中运行，这一环境是超出设计者所预料的。

本书旨在提供坚实的密码学基础知识。密码学的基础包括：范例、方法和抽象化的技巧、定义，以及提供对自然的“安全考虑”的解决方案。我们将提出一些这样的范例、方法和技术，并通过对它们的使用而掌握它们。我们的重点是澄清这些基本概念，并且论证解决这些核心密码问题的可行性。

解决一个密码问题（或说明一种安全需求）是由*定义阶段*（definitional stage）和*构造阶段*（constructive stage）组成的过程。首先，在定义阶段，基于固有利害关系的功能必须标识出来，适当的密码问题也必须被定义。试图列出所有未曾预料的情况是不

可行的，而且容易出错。相反，人们应该按照想象的理想模型中的操作方法定义此功能，然后需要一种候选解决方案在真实且被明确定义的模型（表示对手的能力）中仿真此操作。定义阶段完成后，接着就是构造一个满足此定义的系统。这样的构造可以使用一些更简单的工具，它的安全性可以通过这些工具的性质来证明（当然，实际中这样的机制也必须满足某些特定的效率要求）。

本书的重点在于几个构造型（archetypical）密码问题（例如，加密和签名机制）和几个核心工具（例如，计算难题、伪随机性和零知识证明）。对于每个问题（或工具），我们首先给出其固有的利害关系（或它的直观目标），然后定义这个问题（或工具），最后证明这个问题可以被解决（或这个工具可以被构造）。在最后一步中，我们着重于证明解决此问题的可行性，而不是给出一个切实可行的解决方案。对于给定（或已知）解决方案的实用性（或不实用性）的级别，我们只作为次要问题来讨论。

计算难题

前面提及的特殊构造（和这个领域的大多数构造一样）仅当某种计算困难（即难题）存在时才能存在。特别地，所有这些问题和工具（直接或间接）都要求能生成难题的实例，这一能力在单向函数的定义（进一步讨论参见 2.1 节）中可以获得。于是，单向函数就成为了大多数密码学问题的最小要求。正如我们将要看到的那样，实际上它们对许多密码学问题来说都足够了（其余的可以通过扩充和扩展单向函数存在这一假设来解决）。

仅凭我们目前对有效计算的理解并不能证明单向函数存在。特别地，单向函数的存在性意味着 $\mathcal{NP}$不包含于 $\mathcal{BPP} \supseteq \mathcal{P}$（“大多数”都不是），这一结论能够解决计算机科学方面的许多著名难题。为了证明这一假设的合理性，我们只能给出数百（或数千）位学者的联合信任。而且，这些信任是关于一个简单描述的假设，其有效性基于几个公认的猜想，这些猜想是某些领域的核心（例如，对整数进行因数分解是困难的这一猜想就是计算数论的核心）。

既然无论如何我们都需要一定的假设，那么我们为何不假设我们想要的，即某一固有的密码学问题的解决方案的存在性呢？首先，我们要知道想做什么。如前所述，我们必须首先阐述要做什么，即：我们必须经过典型、复杂的定义阶段，但一旦完成了定义，我们就能假设导出的定义满足要求吗？却却相反。事实是：导出的定义并不意味着它一定能满足，人们可以很容易地定义不存在（没有定义中的明显事实）的对象。要证明一个定义是可行的（因此直观上的安全性条件可以完全满足），方法是构造一个基于更*易于理解*（better-understood）的假设（即更普遍更为广泛认同的假设）。例如，考虑零知识证明的定义，这样的证明（在非平凡的意义上）完全存在并非完全先验的。首先可以通过为二次剩余类命题提出一个零知识证明系统来证明这一概念的非平凡性，（没有附加信息）二次剩余类命题是难于校验的。接着，与先验信任相反，后面将证明单向函数的存在性意味着任意 $\mathcal{NP}$命题都可以用零知识证明。这样，那些根本不知道是否成立的事实（以及那些被认为是错误的事实）都可以通过变形为广泛认同的假设（总之没有它们，许多现代密码学问题就失去依托）来证明其成立。总之，并非所有假设都是等价的，因此把一个复杂的、新的、有疑问的假设转变成一个被广泛认同的简单（或者仅仅是简单）假设是非常重要的。把新任务的解决方案转变成一个已知的简单任务的假定安全方案就意味着：给出一个使用已知原型的构造，可以解决新的任务。这就意味着，我们不但知道（或假设）新任务是可以解决的，而且还能找到一个基于某一原型的解

决方案，这一原型是已知的，并有几个可供选择的实现方法。

结构和先决条件

我们的目标是给出密码学的基本概念、技巧和结论。如前所述，我们的重点在于阐明基本概念及其相互关系。这可以通过一种不依赖于某些通用数论例子的特性的方法来实现。这些特殊的例子在这个领域的发展中起着核心作用，并为所有的密码学原型提供了最切合实际的实现方法，但是这并不表示我们的陈述必须和它们相关联。相反，当我们把概念在抽象层提出，并把它们从特定的实现方法中分离出来时，我们才能以最好的方式阐明它们。与本书关系最紧密的背景知识都是以算法（包括随机化算法）的基本知识、可计算性和初等概率论的方式提出的。某些构造法的实现需要基于（计算性的）数论的背景知识，但是本书并非真的需要这一知识（为了证明本书提出的一些实现方法的实例，本卷在附录中给出了最密切相关的背景知识）。

本著作的构成　本著作由三部分构成（参见图 0.1，本书是第一卷），将分别在三卷中介绍：基本工具、基本应用和在基础之上的知识。第一卷包括一个介绍性的章节和第一部分内容（基本工具），讲述了关于计算难题（单向函数）、伪随机性和零知识证明的知识。这些基本工具将用于第二卷中的基本应用，第二卷是由加密、签名和通用密码协议组成的。

本著作是按逻辑划分为三卷的。而且，这样划分还有如下优点：不必等其他部分完成就可以出版第一卷；同理，我们期望在几年内完成第二卷，然后不必等第三卷完成就可以出版第二卷。

第一卷的结构　第一卷是由一个介绍性的章节（第 1 章）以及关于计算难题（单向函数）、伪随机性和零知识证明的章节（分别对应于第 2~4 章）组成的。另外，还包括两个附录，其中一个是对第 2 章的简单概括。图 0.2 描述了第一卷的主要结构。

第一卷：绪论和基本工具
- 第 1 章：绪论
- 第 2 章：计算复杂性（单向函数）
- 第 3 章：伪随机发生器
- 第 4 章：零知识证明系统

第二卷：基本应用
- 第 5 章：加密机制
- 第 6 章：签名机制
- 第 7 章：通用密码协议

第三卷：在基础之上的知识
- …

图 0.1　本著作的结构

每章结尾给出了一些历史记录、对进一步阅读的建议、一些未决问题和一些习题。习题主要是用来帮助读者理解主要内容和测试其理解程度的，而不是用来检测或鼓励创造力的。未决问题都是相当著名的，我们还建议读者了解一下它们目前的发展状态（例如在我们的即

时更新布告站点上)。

本书的布告站点 我们打算维护一个列出各种类型修正的站点，站点地址是 http://www.wisdom.weizmann.ac.il/~oded/foc-book.html。

第 1 章：绪论
　　本书涉及的主要问题（1.1 节）
　　概率和计算方面的背景知识（1.2 和 1.3 节）
　　严密处理的目的（1.4 节）
第 2 章：计算复杂性（单向函数）
　　动机和定义（2.1 和 2.2 节）
　　单向函数：弱单向隐含了强单向（2.3 节）
　　多样性（2.4 节）和提高性阅读材料（2.6 节）
　　核心断言（2.5 节）
第 3 章：伪随机发生器
　　动机和定义（3.1～3.3 节）
　　基于单向置换的构造（3.4 节）
　　伪随机函数（3.6 节）
　　提高性阅读材料（3.5 节和 3.7 节）
第 4 章：零知识证明
　　动机和定义（4.1～4.3 节）
　　对 $\mathcal{NP}$ 问题的零知识证明（4.4 节）
　　提高性阅读材料（4.5～4.11 节）
附录 A：计算数论中的背景知识
附录 B：第二卷摘要
参考文献

图 0.2 本卷的大致结构

如何使用本书

本书可以用作教科书和参考书。也就是说，它既是面向初学者的又是面向专家学者的。为了达到此目标，本书基础阅读材料的描述是非常详细的，这样大多数计算科学的大学生都能掌握它。程度高的学生（和专家学者）可能会觉得这些部分的进度太慢了，但是我们尝试让他们可以很容易就跳过这些细节部分。特别地，证明过程通常是按一种模式描述的。首先我们给出主要思想的高度概括，然后才开始具体细节的描述。从精练描述到具体细节的转变通常用“详述如下”的字眼来指示。

在有些地方（就像本段这样首行缩进的段落中），我们给出了一些直接而又乏味冗长的细节。还有一些地方（更少），这样的段落给出了与本书的主题关联不大的结论的证明过程。

更高级的阅读材料通常以更快的进度和更少的细节提出。希望我们试图使大范围的读者

都满意，不会不利于任何读者。

教学 本书给出的阅读材料一方面超出了一学期课程所能覆盖的内容，另一方面又缺乏一般人所期望了解的密码学知识。为了调和这些冲突的需求，我们对基础的和高级的阅读材料加以区分，并（在每章的最后一节）对进一步阅读提出建议。特别地，那些标记星号的小节是供高级阅读使用的。

本著作第一卷和第二卷试图提供密码学基础教程所需要的所有阅读材料。对一学期的课程来说，教师一定要跳过所有的高级阅读材料（在目录中用星号标记的），甚至可能需要略过一些基础阅读材料；请参考图 0.3 的建议。根据年级，应该覆盖一个合理层次上的基础阅读材料（例如，所有标记了“要点”的阅读材料和一些“可选”的阅读材料）。第一卷和第二卷也可以用作两学期课程的教科书。不管是哪种情况，第一卷都只覆盖了一学期课程的前半部分，后半部分包含在第二卷中。同时，我们建议后半部分使用其他教材。附录 B 给出了第二卷的简短概括和推荐的其他教材（此外，第二卷中这 3 章的片段和/或初稿分别能在一些早期的文档［99］、［100］和［98］中得到）。

每讲一小时。1～15 讲覆盖第一卷的内容，16～28 讲覆盖第二卷的内容。

第 1 讲：绪论、背景知识等。

　　（视年级而定）

第 2～5 讲：计算复杂性（单向函数）

　　要点：定义（2.2 节），核心断言（2.5 节）

　　可选：弱单向隐含了强单向（2.3 节），以及 2.4.2～2.4.4 节

第 6～10 讲：伪随机发生器

　　要点：定义问题和构造（3.2～3.4 节）

　　可选：伪随机函数（3.6 节）

第 11～15 讲：零知识证明

　　要点：一些定义和构造（4.2.1、4.3.1、4.4.1～4.4.3 节）

　　可选：4.2.2、4.3.2、4.3.3、4.3.4、4.4.4 节

第 16～20 讲：加密机制

　　定义和构造（参考附录 B1.1～B1.2）

　　（加密章节也可以参见手稿［99］的片段。）

第 21～24 讲：签名机制

　　定义和构造（参考附录 B.2）

　　（签名章节也可以参见手稿［100］的片段。）

第 25～28 讲：通用密码协议

　　定义方法和通用构造（概述）。

　　（参考附录 B.3，也可以参见［98］。）

图 0.3　密码学基础的一学期教学计划

虽然一门课程可以只用第一卷作为教材，但是这样的课程不可以作为密码学上的独立课程，因为本卷根本没有涉及加密和签名的基本任务。

实际应用 本书的目标是为密码学提供理论基础。如前所述，这些基础对任何合理的密码学知识的应用来说都是必需的。当然，合理的应用不单单需要理论基础，然而本著作没有给出任何超出理论基础的知识。不过，有了合理的基础，我们就可以学习和评价其他地方（例如［158］）出现的各种有关实际应用的建议。另一方面，如果缺乏合理的基础就不能评判对实际应用的建议，这反过来又会导致不合理的决策。对设计一个能抵抗对手攻击的机制来说，没有什么比误解这些攻击的概念危害更大了。

感谢

首先，我要感谢对我的专业发展有重大影响的 3 位著名人物：Shimon Even，Silvio Micali 和 Shafi Goldwasser。Shimon Even 引导我迈入理论计算科学的殿堂，并密切地领导我迈出第一步。Silvio Micali 和 Shafi Goldwasser 引导我研究密码学基础，并和我分享他们为进一步发展这些基础所做的努力。

我曾与许多研究人员合作过，但是我觉得与 Benny Chor 和 Avi Wigderson 的合作对我的专业发展和事业的影响最大。我要感谢他们，既为他们对我的相关研究的无私奉献，也为和他们一起工作时的兴奋与愉悦。

同时我也要对 Leonid Levin 表示特别的感谢。多年来，我和 Leonid 曾有过许多有趣的讨论，有时我要花很长时间才能意识到这些讨论对我是多么有帮助。

接着，我还要感谢一些同事和朋友，我和他们在密码学和相关问题上彼此有着重要影响。他们包括 Noga Alon，Boaz Barak，Mihir Bellare，Ran Canetti，Ivan Damgard，Uri Feige，Shai Halevi，Johan Hastad，Amir Herzberg，Russell Impagliazzo，Joe Kilian，Hugo Krawcyzk，Eyal Kushilevitz，Yehuda Lindell，Mike Luby，Daniele Micciancio，Moni Naro，Noam Nisan，Andrew Odlyzko，Yair Oren，Rafail Ostrovsky，Erez Petrank，Birgit Pfitzmann，Omer Reingold，Ron Rivest，Amit Sahai，Claus Schnorr，Adi Shamir，Victor Shoup，Madhu Sudan，Luca Trevisan，Salil Vadhan，Ronen Vainish，Yacob Yacobi 和 David Zuckerman。

假如我没有遗漏对我在本书的相关问题上有重要影响的人，对我有恩惠的人是非常多的。其中一定包括许多参考书目中提及的论文的作者，也包括许多我没有引用的有关密码学论文的作者，以及许多有关整个计算理论（本书中理所当然视之为理论）的论文的作者。

最后，我要感谢 Alon Rosen，他仔细审阅了初稿并提出了许多修正建议。

目 录

第 1 章 绪论 …… 1
1.1 密码学：概述 …… 1
1.1.1 加密机制 …… 2
1.1.2 伪随机序列发生器 …… 3
1.1.3 数字签名 …… 3
1.1.4 容错协议和零知识证明 …… 4
1.2 概率论基础知识 …… 6
1.2.1 符号约定 …… 6
1.2.2 3 个不等式 …… 7
1.3 计算模型 …… 9
1.3.1 $\mathcal{P}$, $\mathcal{NP}$ 与 $\mathcal{NP}$-完全 …… 9
1.3.2 概率多项式时间算法 …… 10
1.3.3 非均匀多项式时间算法 …… 12
1.3.4 难处理假设 …… 14
1.3.5 预言机（Oracle Machine） …… 15
1.4 严密处理的目的 …… 15
1.4.1 严密处理的需要 …… 16
1.4.2 严密处理的实际结果 …… 17
1.4.3 保守倾向 …… 18
1.5 其他 …… 19
1.5.1 历史记录 …… 19
1.5.2 关于进一步阅读的建议 …… 20
1.5.3 未决问题 …… 21
1.5.4 习题 …… 21

第 2 章 计算复杂性 …… 23
2.1 单向函数：动机(单向函数的意义) …… 24
2.2 单向函数的定义 …… 25
2.2.1 强单向函数 …… 25
2.2.2 弱单向函数 …… 27
2.2.3 两个有用的长度协议 …… 27
2.2.4 单向函数的候选形式 …… 31

2.2.5 非均匀单向函数 …… 32
2.3 弱单向函数隐含强单向函数 …… 33
2.3.1 定理 2.3.2 的证明 …… 34
2.3.2 一个有趣的例子 …… 37
2.3.3 讨论 …… 38
2.4 单向函数的多样性 …… 39
2.4.1* 通用单向函数 …… 40
2.4.2 单向函数类 …… 41
2.4.3 单向函数类的实例 …… 42
2.4.4 陷门单向置换 …… 44
2.4.5* 无爪（claw-free）函数 …… 46
2.4.6* 关于推荐候选式 …… 48
2.5 核心断言（Hard-Core Predicates） …… 49
2.5.1 定义 …… 49
2.5.2 任意单向函数的核心断言 …… 50
2.5.3* 核心函数 …… 56
2.6* 单向函数的有效放大 …… 59
2.6.1 构造 …… 60
2.6.2 分析 …… 62
2.7 其他 …… 67
2.7.1 历史记录 …… 67
2.7.2 关于进一步阅读的建议 …… 68
2.7.3 未决问题 …… 69
2.7.4 习题 …… 70

第 3 章 伪随机发生器 …… 77

3.1 启发性讨论 …… 78
3.1.1 随机性的计算逼近 …… 78
3.1.2 伪随机发生器的一个严格逼近 …… 78
3.2 计算不可分辨性 …… 79
3.2.1 定义 …… 79
3.2.2 统计相关性 …… 80
3.2.3 重复实验不可分辨性 …… 81
3.2.4* 电路族不可分辨性 …… 84
3.2.5 伪随机总体 …… 85
3.3 伪随机序列发生器定义 …… 85
3.3.1 伪随机发生器的标准定义 …… 85
3.3.2 增加扩展因子 …… 86
3.3.3* 不定长输出的伪随机发生器 …… 90

3.3.4 伪随机发生器的适用性 …… 90
3.3.5 伪随机性和不可预测性 …… 91
3.3.6 伪随机发生器隐含着单向函数 …… 94
3.4 基于单向置换的构造 …… 94
3.4.1 基于单一置换的构造 …… 95
3.4.2 基于置换集合的构造 …… 100
3.4.3* 应用核心函数而不是核心断言 …… 102
3.5* 基于单向函数的构造 …… 103
3.5.1 利用 1-1 单向函数 …… 103
3.5.2 利用正则单向函数 …… 107
3.5.3 在正则单向函数之后的讨论 …… 112
3.6 伪随机函数 …… 113
3.6.1 定义 …… 113
3.6.2 构造 …… 115
3.6.3 应用程序：一个一般的方法论 …… 119
3.6.4* 一般化(普遍化) …… 120
3.7* 伪随机置换 …… 124
3.7.1 一些定义 …… 125
3.7.2 构造 …… 126
3.8 其他 …… 128
3.8.1 历史记录 …… 128
3.8.2 关于进一步阅读的建议 …… 129
3.8.3 未决问题 …… 130
3.8.4 习题 …… 130

第 4 章 零知识证明系统 …… 140

4.1 零知识证明：动机 …… 141
4.1.1 证明的概念 …… 142
4.1.2 获得知识 …… 144
4.2 交互证明系统 …… 145
4.2.1 定义 …… 145
4.2.2 一个实例(*IP*中的图非同构问题) …… 148
4.2.3* *IP*类的结构 …… 151
4.2.4 模型的扩展 …… 152
4.3 零知识证明：定义 …… 152
4.3.1 完备零知识和计算零知识 …… 153
4.3.2 一个例子(*PZK*中的图同构) …… 157
4.3.3 关于辅助输入的零知识 …… 162
4.3.4 零知识证明的顺序合成 …… 164

4.4 *NP*零知识证明 …… 169
4.4.1 承诺方案 …… 170
4.4.2 图着色的零知识证明 …… 173
4.4.3 普遍结论和一些应用 …… 182
4.4.4 二级考虑 …… 185
4.5* 否定结果 …… 187
4.5.1 交互和随机性的重要性 …… 187
4.5.2 无条件结果的限制 …… 188
4.5.3 统计零知识证明的限制 …… 190
4.5.4 零知识和并行合成 …… 190
4.6* 证据不可分辨性和隐藏性 …… 192
4.6.1 定义 …… 193
4.6.2 并行合成 …… 195
4.6.3 构造 …… 196
4.6.4 应用 …… 198
4.7* 知识证明 …… 198
4.7.1 定义 …… 198
4.7.2 减少知识误差 …… 202
4.7.3 *NP*知识的零知识证明 …… 203
4.7.4 应用 …… 203
4.7.5 身份证明(身份认证机制) …… 204
4.7.6 强知识证明 …… 207
4.8* 计算合理性证明(参数) …… 209
4.8.1 定义 …… 210
4.8.2 完备隐藏承诺方案 …… 210
4.8.3 *NP*完备零知识理论 …… 215
4.8.4 多项式对数效率的讨论 …… 216
4.9* 常数轮零知识证明 …… 217
4.9.1 使用完全保密的承诺机制 …… 218
4.9.2 限定欺骗证明者的能力 …… 222
4.10* 非交互零知识证明 …… 225
4.10.1 基本定义 …… 225
4.10.2 构造 …… 226
4.10.3 扩展 …… 230
4.11* 多证明者零知识证明 …… 234
4.11.1 定义 …… 234
4.11.2 两发送者的承诺方案 …… 236
4.11.3 *NP*完备零知识 …… 239
4.11.4 应用 …… 240

4.12 其他 …… 241
4.12.1 历史记录 …… 241
4.12.2 关于进一步阅读的建议 …… 242
4.12.3 未决问题 …… 243
4.12.4 习题 …… 244

附录 A 计算数论背景 …… 250

A.1 素数 …… 250
A.1.1 素数求模的二次剩余 …… 250
A.1.2 在素数求模运算中开方 …… 250
A.1.3 素性检测器 …… 251
A.1.4 素数的均匀选择 …… 252
A.2 合数 …… 252
A.2.1 合数求模的二次剩余 …… 253
A.2.2 合数求模的开方运算 …… 253
A.2.3 勒让德和雅克比符号 …… 254
A.2.4 布卢姆整数和它们的二次剩余结构 …… 254

附录 B 第 2 卷摘要 …… 256

B.1 加密：摘要 …… 256
B.1.1 定义 …… 256
B.1.2 构造 …… 257
B.1.3 防窃听安全性 …… 259
B.1.4 一些建议 …… 261
B.2 签名：摘要 …… 261
B.2.1 定义 …… 262
B.2.2 构造 …… 262
B.2.3 一些建议 …… 264
B.3 密码学协议：摘要 …… 265
B.3.1 定义 …… 265
B.3.2 构造 …… 266
B.3.3 一些建议 …… 267

参考文献 …… 268

绪论

第 1 章

本章 1.1 节首先简单地论述了密码学的主要目标，接着专门论述了有关安全加密术、数字签名和容错协议的基本问题。这些问题又引出了伪随机序列发生器和零知识证明的概念，于是这些概念也随之被讨论。

我们提出的密码学算法是基于计算复杂度的。因此作为介绍性的部分，本章在 1.3 节介绍了一些贯穿本书的计算模块。同样，本章在 1.2 节从本书广泛使用的概率论角度介绍了一些基本背景知识。

最后，在 1.4 节，我们提出了贯穿本书的一些处理方法，并讨论了有关问题。

教学提示：1.4 节的某些部分作为最后一讲（即，作为总结陈述的一部分）可能比作为第 1 讲（即，作为介绍性陈述的一部分）更合适。这一点主要是针对 1.4.2 节和 1.4.3 节来说的。

1.1 密码学：概述

历史上，密码学（cryptography）主要与加密机制（即那些在不安全的通信媒介上提供安全通信的机制）的设计和分析有关。然而，从 20 世纪 70 年代开始，诸如构造不可伪造的数字签名和设计容错协议之类的问题也逐渐成为密码学研究的范围。其实，密码学就是一种与设计一种能抵抗恶意攻击的系统有关的学科。本书提出的密码术基本上都是利用一些在有关著作中都必须阐述的工具实现的，其中最突出的例子就是单向函数（one-way function）、伪随机序列发生器（pseudorandom generator）和零知识证明（zero-knowledge proof）。本节我们将简单地讨论一下这些概念。

首先我们必须声明本书的许多内容都是基于单向函数存在这一假设的。单向函数的定义记录了计算复杂性（computational difficulty）的种类。由于我们提出的密码术试图利用真实世界中攻击者的计算局限性，所以计算复杂性是整个加密方法的内在特性。如果没有计算复杂性，那么这种密码术就失效了。然而，如果像大家广泛认同的那样，不仅各种复杂性存在而且可以有效地生成它们的实例，那么这些复杂性就能产生效用了。可以说“算法上的坏消息”（计算复杂性存在）对密码术来说意味着是好消息。第 2 章我们将用单向函数的形式着重讨论计算复杂性的定义和处理办法。

1.1.1 加密机制

在不安全的媒介上提供安全通信是密码学中最古老和最基本的问题。一个系统是由在某一信道上通信的双方组成的，此信道可能被攻击者（即所谓的窃听者）窃听。通信双方希望交换信息，并且信息尽可能不被窃听者知道。简单地说，加密机制就是一种允许通信双方秘密通信的协议，典型的加密机制由一对算法组成：一个称为加密算法，用于发送方（即发送消息的一方）；另一个称为解密算法，用于接收方。因此，为了发送消息，发送方首先要用加密算法对消息进行加密，再将加密后的消息（我们称之为密文（ciphertext））通过信道发送出去。另一方（即接收方）收到密文时再用解密算法对密文进行解密，就能恢复原始消息（我们称之为明文（plaintext））。

为了使这种机制能够提供保密通信，通信双方（至少是接收方）必须知道一些不为窃听者所知的信息（否则，窃听者就能像接收方那样准确无误地对密文进行解密了）。我们称这种特殊信息为解密密钥（decryption key），它可能依赖于解密算法本身，也可能就是一些参数式和/或解密算法的辅助输入。为了不失一般性，我们可以假设窃听者知道解密算法，但解密算法需要两个输入：密文和解密密钥。需要强调的是：不被窃听者所知的解密密钥的存在性仅仅是进行秘密通信的必要条件。

评价一种加密机制的“安全性”是一件非常棘手的事情，其预备任务就是要理解什么是“安全性”（即凭直觉适当地定义）。已知的定义安全性的方法有两种。第一种（古典）方法是基于信息论的，它与密文中尚存的明文信息量有关。简而言之，如果密文包含明文信息，这种加密机制就被视为是不安全的。研究证明，只有使用的密钥长度至少等于通过此加密算法发送的消息的总长度时，才能保证高度的（即完美的）安全性。所以密钥长度必须大于用它来交换的信息的总长度，这一点极大地限制了此类加密机制的使用，尤其是当有大量信息需要秘密传输时。

第二种（现代）方法也就是本书所遵循的方法，是基于计算复杂度的。它主要建立在如下事实上：密文是否包含明文信息并不重要，重要的是该信息能否有效地被提取出。换句话说，我们并不关心窃听者是否可能提取出特殊信息，而只关心窃听者提取出此信息是否可行。研究证明，这种新的方法（即“计算复杂度”）在密钥长度远小于通过此加密机制发送的消息的总长度时还能保证安全性。例如，你可以用“伪随机序列发生器”（后面我们将对它进行讨论）把较短的密钥扩展成非常长的“伪密钥（pseudo-key）”，使它和同样长度的“真密钥（real-key）”一样安全。

此外，基于计算复杂度的方法允许引入一些在基于信息论的方法中无法成立的概念和原型，典型的例子就是公钥加密机制。前面我们集中讨论了解密算法及其密钥，但是加密算法除了要输入消息外，还要输入一个依赖于解密密钥的辅助输入。这个辅助输入被称为加密密钥（encryption key）。在传统的加密机制及 20 世纪 80 年代前数千年的所有加密机制中，加密密钥都等于解密密钥，这就必须保证窃听者不知道加密密钥，于是密钥分配（key-distribution）问题应运而生了（例如，如何使希望在不安全信道上通信的双方对安全的加密/解密密钥达成一致）。[①] 然而，基于计算复杂度的方法允许窃听者知道加密密钥，但却不会影响此机制的安全

① 传统的解决方案是通过一个安全的替换信道交换密钥，例如，通过护送方式“更昂贵使用（more expensive to use）”。

性。显然，这种机制的解密密钥不等于加密密钥，而且从加密密钥求得解密密钥是不可行的，这种加密机制就是公钥(public-key)加密机制。因为它的加密密钥是可以公开的，所以轻而易举地解决了密钥分配问题。

在专门讨论加密机制的第 5 章(包含在本著作的第二卷中)，我们将讨论私钥加密机制和公钥加密机制，重点放在加密机制安全性的定义上，最后还讨论了如何在各种复杂的假设下构建安全加密机制。其中有些加密机制是建立在伪随机序列发生器的基础上的，伪随机序列发生器将在第 3 章中讨论。还有一些是利用特殊的单向函数(例如 RSA 函数)和/或以某一合数为模的平方运算。

1.1.2 伪随机序列发生器

伪随机序列发生器在加密机制(及其相关机制)的构造中起着重要作用，尤其是能简化私钥加密机制的构造，因此常常在实践中使用(往往是隐含的)。

尽管“伪随机序列发生器”一词在实践中常常使用，但是在密码学和概率类的书中很少有其精确定义。一般说来，在没有明确一个概念的准确定义前就使用它是很危险的，对密码学这种棘手的学科来说更是如此。因此，对密码学来说给出伪随机序列发生器的精确描述是很重要的。

简单地说，伪随机序列发生器就是一种确定性算法，此算法能将较短的随机种子扩展成比它长许多的比特流，这种比特流看起来是“随机”的(尽管实际上并不是)。换句话说，尽管伪随机序列发生器的输出并不真的是随机的，但是要分辨出二者的不同也是不可行的。由于伪随机序列发生器可以基于不同的复杂假设构造，所以伪随机发生器和计算复杂性有很多根本的联系。更进一步说，结论表明，当且仅当单向函数存在时，伪随机序列发生器才存在。

第 3 章专门讨论了伪随机序列发生器。它首先论述了计算的不可分辨性，接着给出了伪随机序列发生器的定义，以及如何使用特殊类型的单向函数(在第 2 章中有定义)来构造伪随机序列发生器。此外，还给出了伪随机函数的定义和构造，它使伪随机序列得到了许多新的应用。

1.1.3 数字签名

数字签名这一概念在非计算机化的世界中是不存在的，然而当引入商务环境下的计算机通信这一概念时，就有必要讨论数字签名了，因为通信双方必须受他们自己制定的建议和/或声明的约束。虽然上几个世纪对“不可伪造的签名”这一问题也有过讨论，但是其讨论对象是手写签名而不是数字签名，而且这些讨论与密码学没有关系。

加密机制和签名方法的“数字化”，以及将计算复杂度的方法引入到信息安全之中，使得加密机制和签名方法相关联成为可能。简单地说，一种不可伪造的签名机制应满足以下几点：

- 每个用户可以有效地生成在他所选的文件上的签名；
- 每个用户能够有效地校验所给的字符串是否是另一(指定的)用户在特定文件上的签名；
- 如果用户本身没有在文件上签名，那么其他任何用户都不能有效地伪造那些用户的签名。

我们对不可伪造的数字签名的简单描述其实也清楚地刻画出了手写签名的本质特点。这些特点分别对应于:每个人具有签名的能力,全体成员达成共识的校验过程,以及人们相信(或确信)用一种可以通过校验的方法伪造签名是不可行的(或者至少是很困难的)。我们很难描述什么程度的手写签名可以满足这些要求,但是我们将要讨论的数字签名对什么程度的数字签名才能满足上述要求有精确的描述。另外,构造不可伪造的数字签名机制的计算前提和构造(私钥)加密机制所使用的计算前提是一样的。

第6章(在本著作的第二卷中)将专门讨论数字签名机制。首先我们将着重给出这些机制的安全性(即不可伪造性)的定义,接着介绍如何在各种极端情况下构造不可伪造的签名机制。此外,我们还讨论了一些有关消息认证的问题。

消息认证

消息认证是与用于加密机制的设置(即,在不安全信道上的通信)有关的工作。假设有一个攻击者监控着信道,可以修改信道上传递的消息,那么通信双方就希望给他们发送的消息加上标识符,使得接收方可以区分被(攻击者)修改的消息与(发送方发送的)原始消息。简单地说,消息认证机制必须满足以下几点:

- 每个通信方都能为它选择的消息有效地生成认证标签;
- 每个通信方都能有效地验证某一字符串是否是某一消息的认证标签;
- 除了通信双方外其他任何一方都不能有效地产生消息认证标签。

从某种意义上说,消息认证类似于数字签名。二者的不同之处在于消息认证系统不要求第三方(可能是不诚实的)验证由指定用户生成的认证标签的有效性,而数字签名系统要求第三方可以校验其他用户生成的签名的有效性。因此,数字签名为消息认证问题提供了一种解决方案。另一方面,消息认证机制并不一定会构成数字签名机制。

数字签名拓宽了密码学的研究范围

把数字签名问题归入密码学拓宽了密码学研究的领域,使之从特殊的安全通信问题扩展到有关限制系统内部或外部成员用"不诚实"的行为窃取信息的各种问题。特别是以下一些问题:

- 在安全通信问题(可利用加密机制解决)中,人们希望尽可能减少窃听者能从两个特定用户间的通信中提取的信息量。在这种情况下,这一特定的系统就由通信双方组成,而窃听者被认为是系统外部("不诚实")的一方。
- 在消息认证问题中,人们期望禁止任何(外部)窃听者修改两个(特定的)用户间的通信。
- 在数字签名问题中,人们期望为系统中的所有用户提供一种方法,此方法能为每个用户生成自我约束的规则,同时保证每个用户不能生成可以约束其他用户的规则。在这种情况下,这一特定的系统就由所有用户组成,潜在的伪造者是系统内部的、不诚实的用户。

因此,广义上说,密码学研究的是如何减少不诚实用户的影响的问题。这些问题一般是通过"容错"(或密码)协议来处理的。

1.1.4 容错协议和零知识证明

有了签名机制人们自然会问什么条件下一方应该给另一方提供签名,这样自然就导致了

密码协议的出现。特别是像互相同时承诺这样的问题(例如,签署合约)就产生了。由商务环境下计算机通信的用户刺激产生的问题则是由“安全实现”协议组成的(例如,实现安全的、公正的投票)。

并发问题

一个典型的并发问题就是在某些特殊场合下签署合约时需要同时交换机密信息。一个并发机密信息交换系统由掌握着“机密信息”的双方组成。我们的目标就是要制定一个协议,使得双方都正确地遵循此协议时,最终都能获得对方的机密信息,在任何情况下(即使一方有欺骗行为),第一方可获得第二方的机密信息,当且仅当第二方获得了第一方的机密信息。理想的并发机密信息交换只有假设在一定程度上可以信任的第三方存在才能实现。事实上,如果有可信任的第三方的积极参与,并发机密信息交换的实现是很容易的:每一方都(利用安全的信道)把各自的机密信息发送给可信任的第三方,第三方收到双方机密信息时再把第一方的机密信息发送给第二方,第二方的机密信息发送给第一方。但是这种方案存在以下两个问题:

1. 此方案需要“外部”一方在所有场合(包括通信双方都很诚实的情况)的积极参与,而其他解决方案要求外部方更少地参与。

2. 此方案要求一个完全可信任的第三方实体存在。然而在某些应用场合,这样的实体并不存在。不仅如此,我们还要讨论如何利用多数诚实的用户(尽管诚实的用户的身份可能并不知道)来实现可信任的第三方的问题。

函数和可信任方的安全实现方案

另一种类型的协议问题是关于函数的安全实现问题。如果把问题特殊化,就是要讨论如何评估具有多个本地输入值(其中每个输入值是由不同用户控制的)的函数。一个典型的例子就是投票,此函数的功能是要实现大多数人同意。用户 A 的本地输入值就是一个比特,它代表 A 的选票(例如“反对”或“支持”)。简单地说,安全评估一个特殊函数的协议必须满足以下几点:

- 保密性:任何人单靠函数值都无法获得其他人的输入信息。
- 健壮性:任何人单凭改变自己的输入值都不能影响函数值。

有时,当少数用户达成共谋时,也要求上述条件成立。

显然,如果已知有一个完全值得信赖的用户,就可以用一个简单的方法来解决任何函数的安全性评估问题。每个用户只需通过安全的信道把他的输入信息发送给可信任的那个用户,该用户收到所有用户的输入后,计算此函数值,再把结果反馈给所有用户,并从其存储器中删除所有的中间计算过程(包括其收到的输入值)。然而现实情况中,肯定不可能对一个用户达到这种程度的信任(例如他能自动抹去所有他所获得的信息)。这样,函数安全性评估问题又导致了如何寻找可信任方这一问题的出现。研究表明,可信任方可以通过多数诚实的用户(尽管诚实的用户的身份可能并不知道)来实现。这个结论是这一领域的主要成果,在本著作第二卷的第 7 章中我们将着重仿真和证明这个结论(包括在不同变量下的情况)。

范例:零知识证明

构造密码协议的主要工具是零知识证明系统以及零知识证明系统在所有 $\mathcal{NP}$(假设单向函数存在)代码中存在的事实。简而言之,零知识证明除了能验证断言的有效性外没有其他任何作用。零知识证明为“迫使”每一方遵循给定的协议提供了一个有效的工具。

下面我们来简单描述零知识证明的作用：假设系统中名叫 Alice 的一方收到了来自 Bob 的已加密的消息，接着她要把此消息的最低有效位（the least sighificant bit）发送给 Carol。然而，如果 Alice 仅发送这个最低有效位（the least sighificant bit），Carol 就无法判断 Alice 是否有欺骗行为。虽然 Alice 可以把整个消息及其解密密钥展示给 Carol 以证明她没有欺骗 Carol，但是这样做展示给 Carol 的信息大大超过了 Carol 本来应该获得的信息。所以更好的办法就是让 Alice 给发给 Carol 的比特增加零知识证明来证实这个比特的确是消息的最低有效位）。需要强调的是上述结论是在"*NP* 类型"条件下成立的（因为前面给出的证据可以有效地被验证），因此，在 *NP* 论题下零知识证明的存在性就意味着不用展示任何多余的信息就可以证明上述结论。

第 4 章以零知识证明为主，着重阐述了上述结论（即，在任何 *NP* 论题下构造零知识证明）。此外，我们还将讨论有关零知识证明的许多变量和有关此问题的各个方面，以及它们对零知识证明适用性的影响。

1.2 概率论基础知识

概率论是密码学中的关键理论知识，在讨论信息或信息掩藏（即保密）的时候概率论尤为重要。我们假定本书的读者已熟悉概率论的基本概念，本节中我们仅列举那些贯穿本书的符号以及 3 个有用的概率不等式。

1.2.1 符号约定

全书只涉及离散型概率分布。典型的情况是概率空间由一个集合构成，该集合中所有的字符串都服从均匀分布并具有确定长度 l。换言之，这个样本空间是一个长度为 l 比特的字符串集合，且取值每个字符串的概率是 2^{-l}。传统地，我们将从样本空间至实数域的变换称为随机变量，这里我们扩大一下此标准术语的含义，即当存在从样本空间至二进制字符串集合的变换时，我们也可以借用术语随机变量。通常我们并不指定概率空间，而是直接讨论随机变量。举例来说，我们可以假定 X 是一个定义于所有字符串集合上的随机变量，使得 $\Pr[X=00]=\frac{1}{4}$，$\Pr[X=111]=\frac{3}{4}$（这种随机变量可以定义在取样空间 $\{0,1\}^2$ 之上，使得 $X(11)=00$，$X(00)=X(01)=X(10)=111$）。大多数情况下，概率空间是由具有特定长度的所有字符串构成的；典型情况下，这些字符串代表着某种随机化过程的随机选择结果（参见下一节），随机变量就是此随机化过程的输出。

如何阅读概率论题 我们所有的概率论题均针对预先定义的随机变量的函数。特别地，我们将给出 $\Pr[f(X)=1]$ 的概率，式中 X 是事先定义的随机变量（而 f 是一个函数）。还有一个重要的约定就是：*一个给定的符号在所有概率论题中均表示相同的（惟一的）随机变量*。因此，如果 $B(\cdot,\cdot)$ 是一个取决于两个随机变量的布尔表达式且 X 是一个随机变量，那么 $\Pr[B(X,X)]$ 就表示当 x 以给定概率 $\Pr[X=x]$ 出现时 $B(x,x)$ 所具有的概率值。也即，

$$\Pr[B(X,X)]=\sum_x \Pr[X=x]\cdot\chi(B(x,x))$$

式中 χ 是一个指标函数，如果事件 B 成立则 $\chi(B)=1$，否则为 0。举例来说，对每个随机变量 X，总有 $\Pr[X = X]=1$ 成立。这里我们强调指出的是，如果 x，y 相互独立且具有相同的概率分布，那么要讨论 $B(x, y)$ 的概率，就必须定义两个独立的随机变量，这两个随机变量具有相同的概率分布。因此，若 X，Y 是两个独立的随机变量，则 $\Pr[B(X, Y)]$ 就表示当一组 (x, y) 以给定概率 $\Pr[X = x]\cdot\Pr[Y = y]$ 出现时 $B(x, y)$ 所具有的概率。

也即，

$$\Pr[B(X,Y)] = \sum_{x,y}\Pr[X = x]\cdot\Pr[Y = y]\cdot\chi(B(x,y))$$

举例来说，对任意两个独立随机变量 X，Y，只有当 X 与 Y 都是平凡（即将全部概率值赋予一个字符串）时，才有 $\Pr[X = Y]=1$。

几种典型的随机变量 全书中 U_n 均指均匀分布在长度为 n 的字符串集合上的随机变量。也即，如果 $\alpha\in\{0,1\}^n$，则 $\Pr[U_n=\alpha]=2^{-n}$，否则为 0。此外，我们有时也将使用到（任意地）分布于 $\{0, 1\}^n$ 或者 $\{0, 1\}^{l(n)}$ 的随机变量（l 是一个 $\mathbb{N}\to\mathbb{N}$ 的变换），这样的随机变量通常被标记为 X_n，Y_n，Z_n 等。我们强调指出的是，某些情况下 X_n 分布于 $\{0, 1\}^n$，而在另一些情况下分布于 $\{0, 1\}^{l(n)}$，$l(\cdot)$ 是一种变换，它通常是一个多项式。还有一种随机变量，即随机算法作用于一个定值输入得到的输出，我们将在 1.3 节中对其进行讨论。

1.2.2 3 个不等式

以下讨论的不等式在整本书中都将十分有用。所有的不等式均依据实值随机变量定义。最基本的不等式是马尔科夫（Markov）不等式，它指出，对于具有最大值或最小值的有界随机变量，在一个偏离随机变量的数学期望的数值与此随机变量被指定为这个数值的概率之间，必定存在某种关系。特别地，定义随机变量 X 的数学期望为 $\mathsf{E}(X)\stackrel{\text{def}}{=}\sum_v\Pr[X=v]\cdot v$，我们得到以下不等式。

Markov 不等式：令 X 为一非负随机变量，v 为一实数，则有 $\Pr[X\geq v]\leq\frac{\mathsf{E}(X)}{v}$，等价地，有 $\Pr[X\geq v\cdot\mathsf{E}(X)]\leq\frac{1}{v}$。

证明：

$$\begin{aligned}\mathsf{E}(X) &= \sum_x\Pr[X = x]\cdot x \\ &\geq \sum_{x<v}\Pr[X = x]\cdot 0 + \sum_{x\geq v}\Pr[X = x]\cdot v \\ &= \Pr[X\geq v]\cdot v\end{aligned}$$

得证。

Markov 不等式常用于不了解随机变量分布的情况，它只要求了解随机变量的期望和至少一个在它的取值范围内的界。参见习题 1。

利用 Markov 不等式可以得到一个随机变量偏离其均值"更紧"的界。如果能够得到关于随机变量的额外知识（特别地，一个随机变量关于其方差的好的上界），那么这个称为契比雪夫

(Chebyshev)不等式的界将十分有用。对于一个数学期望有界的随机变量 X，定义 X 的方差为 $\mathsf{Var}(X)\stackrel{\text{def}}{=}\mathsf{E}[(X-\mathsf{E}(X))^2]$，注意到 $\mathsf{Var}(X)=\mathsf{E}(X^2)-\mathsf{E}(X)^2$。

Chebyshev 不等式：令 X 为一随机变量，且 $\delta>0$，则有

$$\Pr[|X-\mathsf{E}(X)|\geq\delta]\leq\frac{\mathsf{Var}(X)}{\delta^2}。$$

证明：定义一个随机变量 $Y\stackrel{\text{def}}{=}(X-\mathsf{E}(X))^2$ 并利用 Markov 不等式，可得：

$$\begin{aligned}\Pr[|X-\mathsf{E}(X)|\geq\delta]&=\Pr[(X-\mathsf{E}(X))^2\geq\delta^2]\\&\leq\frac{\mathsf{E}[(X-\mathsf{E}(X))^2]}{\delta^2}\end{aligned}$$

结论成立。

Chebyshev 不等式对于通过重复取样进行的逼近的错误概率分析尤其有用，它只要求假定取样是以两两独立的方式进行的。

推论(两两独立取样)：令 $X_1,X_2,...,X_n$ 是两两独立的随机变量，具有相同的数学期望(记为 μ)和相同的方差(记为 σ^2)，则对于任一 $\varepsilon>0$，都有：

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n}-\mu\right|\geq\varepsilon\right]\leq\frac{\sigma^2}{\varepsilon^2 n}$$

若对于任何 $i\neq j$ 以及任意的 a 和 b，均有 $\Pr[X_i=a\wedge X_j=b]=\Pr[X_i=a]\cdot\Pr[X_j=b]$，则称 X_i 是两两独立的。

证明：定义随机变量 $\overline{X}_i\stackrel{\text{def}}{=}X_i-\mathsf{E}(X_i)$，注意到 $\overline{X}_i$ 是两两独立的且具有零数学期望。对该随机变量 $\sum_{i=1}^n\frac{X_i}{n}$ 利用 Chebyshev 不等式并使用数学运算的线性性质，可得：

$$\begin{aligned}\Pr\left[\left|\sum_{i=1}^n\frac{X_i}{n}-\mu\right|\geq\varepsilon\right]&\leq\frac{\mathsf{Var}\left[\sum_{i=1}^n\frac{X_i}{n}\right]}{\varepsilon^2}\\&=\frac{\mathsf{E}\left[\left(\sum_{i=1}^n\overline{X}_i\right)^2\right]}{\varepsilon^2\cdot n^2}\end{aligned}$$

既然有(再次使用数学期望的线性性质)$\mathsf{E}[(\sum_{i=1}^n\overline{X}_i)^2]=\sum_{i=1}^n\mathsf{E}[\overline{X}_i^2]+\sum_{1\leq i\neq j\leq n}\mathsf{E}[\overline{X}_i\overline{X}_j]$，利用 $\overline{X}_i$ 的两两独立条件，得 $\mathsf{E}[\overline{X}_i\overline{X}_j]=\mathsf{E}[\overline{X}_i]\cdot\mathsf{E}[\overline{X}_j]$，再联系到 $\mathsf{E}[\overline{X}_i]=0$，有：

$$\mathsf{E}\left[\left(\sum_{i=1}^n\overline{X}_i\right)^2\right]=n\cdot\sigma^2$$

推论成立。

利用两两独立取样，逼近的错误概率会随着取样点数线性递减。使用完全独立的取样点，逼近过程中的错误概率可以随着取样点数呈现指数下降(若对于任意序列 $a_1,a_2,...,a_n$，均有 $\Pr[\wedge_{i=1}^n X_i=a_i]=\prod_{i=1}^n\Pr[X_i=a_i]$，则称随机变量 $X_1,X_2,...,X_n$ 是完全独立的)。支持以上陈述的概率界将在下面给出。第一个界通常被称为切诺夫(Chernoff)界，主要适用于0-1随机变量(即随机变量的值只可取 0 或 1)。

Chernoff 界：令 $p\leq\frac{1}{2}$ 且 $X_1,X_2,...,X_n$ 是独立的 0-1 随机变量，对任意 i 都有 $\Pr[X_i=1]=p$，则对于所有的 ε，$0<\varepsilon\leq p(1-p)$，都有：

$$\Pr\left[\left|\frac{\sum_{i=1}^{n} X_i}{n} - p\right| > \varepsilon\right] < 2 \cdot e^{-\frac{\varepsilon^2}{2p(1-p)} \cdot n}$$

通常我们使用这个界时令 p 为一常数 $p\approx\frac{1}{2}$，这种情况下，n 个独立的取样在逼近时偏离数学期望达 ε 的概率值 δ 将随 $\varepsilon^2 n$ 呈指数递减。这种逼近被称为 (ε,δ) 逼近，它可以通过 $n=O(\varepsilon^{-2}\cdot\log(1/\delta))$ 个取样点达到。需要记住的是，充分的取样点数是与 ε^{-1} 的多项式级关联且与 δ^{-1} 的对数级关联的。因此，使用 poly(n) 个取样，误差概率（也即 δ）可以忽略不计（作为 n 的函数），但是估计的准确度（也即 ε）只需要任意确定的多项式分数就可以确定上界（而不可能做到忽略不计）[①]。需要强调的是，取样数对于 ε 的依赖程度并不比两两独立的取样情况强，完全独立取样的优势只在于取样数对于 δ 的依赖性。

还有一个更加一般的界，适用于一般随机变量（并不一定服从 0-1 分布）的数学期望的逼近，在下面给出。

Hoefding 不等式：[②]令 $X_1, X_2, \ldots, X_n$ 是具有相同概率分布的独立随机变量，每个变量都在（实）区间 $[a, b]$ 上取值，用 μ 表示每个变量的数学期望值，则对每个 $\varepsilon>0$，都有：

$$\Pr\left[\left|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\right| > \varepsilon\right] < 2 \cdot e^{-\frac{2\varepsilon^2}{(b-a)^2} \cdot n}$$

Hoefding 不等式在估计定义域是很多个取值的集合的函数的均值时十分有用，特别是当希望的错误概率可以忽略不计时。如果可以有效地对集合取样并且按可能（函数的）取值有界时，我们就能利用这个不等式。参见练习 2。

1.3 计算模型

密码学的研究很大程度上取决于计算复杂度。因此，谈论密码学时有必要对计算复杂度的相关背景知识作一定了解。本节中，我们将简要地回顾各种复杂性类型，包括 ***P***，***NP***，***BPP*** 和非一致 ***P***（即 ***P***/poly），以及预言机（oracle machine）的概念。此外，我们还将讨论本书余下部分使用到的难处理假设的类型。

1.3.1 *P*，*NP* 与 *NP*-完全

传统的计算设备通常将有效的计算与 ***P*** 复杂度相关联。回想前面本书中给出的方法更多的是书面形式的，以使计算设备可以随机化。

定义 1.3.1（复杂度类 *P*）：如果存在一种确定性的图灵机 M 和一个多项式 $p(\cdot)$，并且满足：

- 当输入字符串 x 时，M 机器至多在 $p(|x|)$ 步以后停止；
- $M(x)=1$ 当且仅当 $x\in L$。

---

① 此处以及本书的余下部分用 poly() 代表某个固定却未确定的多项式。

② 一个更加一般的形式要求 X_i 相互独立，但不一定等同分布，使用 $\mu \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}(X_i)$，参见文献[6, app. A]。

则称 **P** 是在(确定性的)多项式时间内可识别的一类语言。

类似地,复杂度类 **NP** 与那些有解的计算难题相关联,如果有解,则可以有效地测试其正确性。惯例上将 **NP** 定义为可被非确定性的多项式时间图灵机识别的语言类型。更加基本的 **NP** 的描述由以下等价的定义给出。

定义 1.3.2(复杂度类 NP):如果存在布尔关系 $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ 与多项式 $p(\cdot)$,且 R_L 在(确定性的)多项式时间内可识别,同时 $x \in L$ 当且仅当存在 y 满足,$|y| \leq p(|x|)$,$(x,y) \in R_L$,则称语言 L 是属于 **NP** 类的。同时称此 y 是 $x \in L$ 成员资格的证据。

因此,**NP** 由这样一种语言构成,即存在可被有效验证的、简要的用以证明成员资格的证据。大多数观点认为 $\boldsymbol{P} \neq \boldsymbol{NP}$,毫无疑问,这个论题的解决是计算机科学界最能引起争论的公开问题。如果真的是 $\boldsymbol{P} \neq \boldsymbol{NP}$,那么就会存在一种语言 $L \in NP$,使得每一种可识别 L 的算法在最坏的情况下都需要超多项式的运行时间。当然,所有的 **NP**-完全语言(接下来定义)在最坏的情况下都将具有超多项式时间的复杂度。

定义 1.3.3(NP-完全):如果一种语言是 **NP** 的,且每一种 **NP** 语言关于它都是多项式可约的,那么这种语言就是 **NP**-完全的。语言 L 关于语言 L' 是多项式可约的,如果存在一个多项式时间内可计算的函数 f 使得 $x \in L$ 当且仅当 $f(x) \in L'$。

目前所知的 **NP**—完全语言包括:(命题规则的)可满足性,图的着色,图的汉密尔顿回路。

1.3.2 概率多项式时间算法

随机化算法在密码学中起着关键作用。这些算法用于允许合法成员生成密文,同时攻击者也可使用。本书的读者应熟悉并掌握这类算法。

1.3.2.1 随机化算法的一个例子

为了例证随机化算法的概念,这里给出一个简单的随机化算法,以确定是否一个给定的(无向)图是连通的(也即图中任意一对顶点之间均存在路径)。下面的算法是有趣的,因为这个算法与标准的(基于 BFS、DFS)算法相比可以大大节约空间。

验证一个图是否是连通的,可以很容易地简化为验证任意一对顶点间的连通性。[①] 这样我们就将任务集中为确定在给定的图中两个顶点是否是相连的。

算法 输入图 $G=(V,E)$ 以及两个顶点 s 和 t,从顶点 s 出发,随机前进的步长为 $O(|V|\cdot|E|)$,每一步都判断是否遇到顶点 t。如果经过顶点 t,那么此算法将接受;否则将不接受。随机前进的意义在于,每一步我们都随机选取一条当前顶点的邻接边,并通过这条边到达另一个顶点。

分析 很明显,如果图 G 中 s 与 t 没有连接,那么上述算法接受的概率将是零。更困难的分析在于证明如果图 G 中 s 与 t 存在连接,那么此算法接受的概率将至少是 $\frac{2}{3}$(证明留至习题 3)。因此,无论是哪种情况,这个算法的错误概率至多是 $\frac{1}{3}$。错误概率还可通过多次启动算法

---

① 这种简化的空间复杂度是很低的,我们只需存储(当前验证的)两顶点的名称。然而,时间复杂度却相对较高。我们必须触发验证过程 $\binom{n}{2}$ 次,n 是图中的顶点数目。

而进一步降低(每一次启动算法都使用新的随机选择)。

1.3.2.2 随机化算法的两种观点

随机化算法(机器)可以用两种等价的方法来理解。第一种方法就是允许算法进行随机移动(即抛硬币)。严格来说,可以用一个图灵机的模型来表示之,这个模型中的迁移变换将原先的二元序对形式(⟨state⟩,⟨symbol⟩)映射为三元序对形式(⟨state⟩,⟨symbol⟩,⟨direction⟩)。图灵机的下一步就取决于这些三元序对的随机选择。也即,移动一步时,机器需要随机选择(每种可能各有$\frac{1}{2}$概率),要么是第一个三元序对,要么是第二个三元序对,然后依此移动。这些随机选择被称为机器内部的抛硬币。输入 x 时,概率机 M 的输出不是字符串,而是一个假定字符串为可能值的随机变量。这个随机变量记作 $M(x)$,是由 M 的内部抛硬币归纳而得。$\Pr[M(x)=y]$表示在输入为 x 时机器输出 y 的概率。相应的概率空间就是对内部抛硬币取均匀概率分布时的所有可能结果[①]。因为我们考虑的只是多项式时间的机器,因此为了不失一般性,我们可以假定在输入为 x 时机器 M 抛硬币的次数取决于抛硬币的结果并记为$t_M(x)$。将输入为 x 时机器 M 抛硬币的结果为 r 的输出记作 $M_r(x)$。这样,$\Pr[M(x)=y]$就仅仅是 $M_r(x)=y$ 时与 $r\in\{0,1\}^{t_M(x)}$ 的比值。也就是:

$$\Pr[M(x) = y] = \frac{|\{r \in \{0,1\}^{t_M(x)} : Mr(x) = y\}|}{2^{t_M(x)}}$$

理解随机化算法的第二种方法就是将机器的内部抛硬币视作一种辅助输入。也就是说,我们考虑一种具有双输入的确定性机器。第一种输入就是第一种方法里的"真实的输入"(也即 x),第二种输入就是表示内部抛硬币的可能输出序列,这样我们得到一种与前述记号 $M_r(x)$相对应的记法 $M(x,r)$。在第二种方法中,考虑 $M(x,r)$的概率分布时,对于任何固定的 x,r 在区间$\{0,1\}^{t_M(x)}$上均匀分布。更具体点说,这里的抛硬币不是"内部的",而是在被附加了"外部"抛硬币设备的机器中进行的。

在继续下面的章节以前,我们在这里有必要提醒大家不要将"非确定性"机器的虚构模型与概率机器模型相混淆。前者是一种非现实的模型,主要适用于讨论那些其结论能够被有效验证的问题(如 **NP** 的定义),而后者是一种真实的计算模型。

本书如无特别声明,概率多项式时间图灵机是指一种概率机器,这种机器总是经过多项式数目(输入长度的多项式)的步骤之后停止(与内部抛硬币的结果是独立的)。于是概率多项式时间机器 M 的抛硬币的数目将被一个输入长度的多项式所限制,记作 T_M。最后为了不失一般性,我们假定输入为 x 时机器总是进行 $T_M(|x|)$次抛硬币。

1.3.2.3 联系 *BPP* 与"有效的"计算

我们讨论的基本命题就是将"有效的"计算与概率多项式时间的计算这二者联系起来。也就是说,我们只认为那些运算时间被输入长度的多项式限制的随机化算法(也即概率图灵机)

---

① 这个句子比它初看起来要更加复杂些。简单的情况是当输入 x 时机器 M 总是进行相同次数的内部抛硬币(彼此结果间独立),一般的情况是,抛硬币的结果可能取决于上一次抛硬币的结果。对每一个 r,如果结果序列是 r 的严格前缀时机器不中断,则抛硬币序列结果为 r 的概率是 $2^{-|r|}$,否则就是 0。幸运的是,因为我们考虑的是多项式时间的机器,所以我们就能够修改所有的机器以满足简单情况下的结构(从而避免了上述的复杂性)。

是有效的。

命题：有效的计算对应着可被概率多项式时间图灵机完成的计算。

针对这种计算的复杂度类就是一种（以高概率）可被概率多项式时间图灵机理解的语言类，记作 ***BPP***。这里的概率主要取决于机器对字符串 x 能做出正确判决的事件。

定义 1.3.4（有界概率多项式时间，*BPP*）：如果

- 对于每一个 $x \in L$，$\Pr[M(x)=1] \geq \frac{2}{3}$ 成立，且
- 对于每一个 $x \notin L$，$\Pr[M(x)=0] \geq \frac{2}{3}$ 成立。

则称 L 是可被概率多项式时间图灵机 M 识别的，

BPP 就是可被概率多项式时间图灵机（即随机化算法）识别的语言类。

"有界概率"意指成功概率要大于 $\frac{1}{2}$，事实上，将定义 1.3.4 中的常数 $\frac{2}{3}$ 更换为任何其他大于 $\frac{1}{2}$ 的常数都不会改变定义的类，参见习题 4。类似地，常数 $\frac{2}{3}$ 如果被 $1-2^{-|x|}$ 取代，该类也不变，参见习题 5。结论就是 ***BPP*** 中的语言可以被概率多项式时间图灵机识别，而错误概率可以忽略不计。我们使用忽略不计的意图在于描述任何比某个多项式的倒数下降得还要快的变换。

可忽略不计的函数

定义 1.3.5（可忽略不计）：如果对任一正多项式 $p(\cdot)$，存在 N，当 $n>N$ 时满足

$$\mu(n) < \frac{1}{p(n)}$$

则称函数 $\mu:\mathbb{N} \to \mathbb{R}$ 是可忽略不计的。

例如，函数 $2^{-\sqrt{n}}$ 与 $n^{-\log_2 n}$ 都是可忽略不计的（是 n 的函数）。当与确定的多项式相乘时，这些函数的可忽略不计的性质不变。也就是说，对每一个可忽略不计的变换 μ 与任意多项式 p，函数 $\mu'(n) \stackrel{\text{def}}{=} p(n) \cdot \mu(n)$ 也是可忽略不计的。于是，一个以可忽略不计的概率发生的事件几乎不可能发生，即使将这个实验重复进行多项式次。

约定　定义 1.3.5 中我们提到"存在 N，当 $n>N$ 时"，今后我们将使用"对任何足够大的 n"这样更简明扼要的提法。这就使得一个限定语（即 $\exists N$）成为隐含的，这在那些含有多个（更重要的）限定语的声明中尤其有用。

1.3.3　非均匀多项式时间算法

一种更强的（但不太现实的）有效计算模型是非均匀多项式时间算法模型，这种模型只能用做消极防御，也就是说，即使真有这种模型机，也做不了什么。显然，如果攻击者采用这样的模型机，同样不会造成危害。

非均匀多项式时间算法实际是一个二元组 $(M,\bar{a})$，其中 M 是一个双输入多项式时间算法图灵机，$\bar{a}=a_1,a_2,\ldots$ 是一个无限序列，且 $|a_n|=\text{poly}(n)$。[①] 下面我们以二元组 $(x,a_{|x|})$ 为输入，考虑对应于每一个 x，图灵机 M 的计算方法。直观上看，a_n 可认为是从"外部"供给的额外"信息"（与 $x\in\{0,1\}^n$ 一起被输入）。这里我们强调对于所有同样长度（即 n）的输入，算法机能够得到同样的"信息"（即 a_n）。直观上看，信息 a_n 在某些场合（即输入长度为 n 的一些计算

① poly() 表示一些确定的多项式；即存在一些多项式 p 使得 $|a_n|=p(n)$，对所有的 $n\in\mathbb{N}$。

中)可能是有用的,但是对所有可能的 2^n 个输入都编码是不可能的。

另一种理解非均匀多项式时间算法机的方法是考虑一个图灵机无限序列, $M_1, M_2, \ldots$, 使得当输入长度为 n 的序列时, M_n 描述的长度和它的运行时间都受 n 的多项式的限制(对于所有序列, n 是固定的)。算法机 M_n 只用于输入序列长为 n 的场合。要注意两种理解非均匀多项式时间算法的方法之间的联系。由对 $(M, (a_1, a_2, \ldots))$ (第一种定义)产生了一个图灵机无限序列 $M_{a_1}, M_{a_2}, \ldots$, 这里 $M_{a_{|x|}}(x) \stackrel{\text{def}}{=} M(x, a_{|x|})$。另一方面,正如第二种定义所说,序列 $M_1, M_2, \ldots$ 又产生了一个序对 $(U, (<M_1>, <M_2>, \ldots))$, 这里 U 是一般的图灵算法机, $\langle M_n \rangle$ 是算法 M_n 的描述(即 $U(x, \langle M_{|x|} \rangle) = M_{|x|}(x)$)。

在 1.3.3 节的第一句中,我们就指出非均匀多项式时间算法是比概率多项式时间算法更强的一种模型,这种陈述在许多情况下都有效(后面的定理 1.3.7 中代码识别就是一例)。特别地,在本书中我们所讨论的所有内容中它都是有效的,所以我们有下面简单的"亚定律"。

亚定律:一切能通过概率多项式时间算法获得的结果都能通过非均匀多项式时间算法机获得。

如果想到掷硬币的实验,就会发现亚定律明显是错误的,因此我们不能简单地从字面上去理解它,它是告诉我们一个通过推理证明的真理,例如我们可想想代码识别的前后过程。

定义 1.3.6:非均匀多项式时间算法的复杂性(记作 $\boldsymbol{P}$/poly)是可以被多项式时间算法产生的非均匀序列识别的语言 L 的类,即如果存在一个图灵机无限序列 $M_1, M_2, \ldots$ 满足:

1. 存在一个多项式 $p(\cdot)$,使得对于每一个 n,描述 M_n 的长度都以 $p(n)$ 为上界。

2. 存在一个多项式 $q(\cdot)$,使得对于每一个 n,输入长为 n 的序列后 M_n 的运行时间都以 $q(n)$ 为上界。

3. 对于每一个 n,全部 $x \in \{\mathbf{0}, \mathbf{1}\}^n$, M_n 接受 x 当且仅当 $x \in L$。

若满足上述 **3** 点,则 $L \in \boldsymbol{P}$/poly。

注意,非均匀性是在不考虑序列机的结构的条件下隐含的,只对这些算法机的存在性做要求。相比之下,如果我们依赖多项式时间算法的存在性来扩展定义 **1.3.6**(这种算法一元地输入 $\mathbf{1}^n$,输出 M_n 的描述),那么我们得到的是笨拙的 $\boldsymbol{P}$ 的定义。另一方面,显然 $\boldsymbol{P} \subseteq \boldsymbol{P}$/poly(实际上,严格包含可通过非递归的一元文字来证明)。此外:

定理 1.3.7: $\boldsymbol{BPP} \subseteq \boldsymbol{P}$/poly。

证明: M 是一个概率多项式时间的、用于识别 L 的图灵机, $L \in \boldsymbol{BPP}$。定义 $\chi_L(x) \stackrel{\text{def}}{=} 1$ $(x \in L)$, $\chi_L(x) \stackrel{\text{def}}{=} 0$(其他)。则对所有 $x \in \{0,1\}^*$,都有

$$\Pr[M(x) = \chi_L(x)] \geq \frac{2}{3}$$

不妨设对每一次长度为 n 的输入,图灵机 M 使用相同的数据,记作 $m = \text{poly}(n)$,对于掷硬币试验, $x \in \{0,1\}^n$,显然对所有 $x \in \{0,1\}^n$,都有一投掷序列 $r \in \{0,1\}^m$ 使得 $M_r(x) = \chi_L(x)$(事实上,大多数序列 r 有此性质)。然而是否一个序列能满足所有 $x \in \{0,1\}^n$ 中的 x? 大概是不行的(请举例说明!)。尽管如此。我们可找到 $r \in \{0,1\}^m$ 中 r 序列满足 $\frac{2}{3}$ 的长为 n 的 x(认为如果 $\frac{2}{3}$ 的 r 序列对每个 x 都满足,那么一定有一个 r 满足至少是 $\frac{2}{3}$ 的 x 序列),然而,这并没有给我们一个适合所有 $x \in \{0,1\}^n$ 的 r。想得到这样一个 r,我们必须使用前述有关算法机 M' 的错误概率呈指数幂减小的结论。这样的算法机肯定存在,这在习题 5 中将得到说明。

也就是说，对所有 $x\in\{0,1\}^*$，

$$\Pr[M'(x)=\chi_L(x)]>1-2^{-|x|}$$

应用普遍性证明，我们得出结论：存在 $r\in\{0,1\}^m$，用 r_n 表示，满足 $\{0,1\}^n$ 中的至少 $1-2^{-n}$ 倍的 x 序列。那么 r_n 满足所有 2^n 个长为 n 的输入序列。图灵机 M'（实数双输入机）与前面构造的无限序列 $r_1,r_2,\cdots$ 一起说明了 $L\in$ $\mathcal{P}$/poly。

非均匀线路族　研究非均匀多项式算法更方便的一种方式（实际上也是本书使用的方式）是多项式长度的布尔线路族，布尔线路是一有向无循环图，每一内部节点以{∧，∨，¬}中一个元素为标号¬）。没有入边的节点称为输入节点，没有出边的节点称为输出节点。标有¬的节点只能有一个子项。线路上的计算方法是：首先将输入比特放置到输入节点上（一节点一位），其余步骤如下。如果一个入度为 d 的标有∧的节点的孩子的值为：$v_1,v_2,...v_d$，则这个节点有值 $\wedge_{i=1}^{d}v_i$。类似地，对标有∨和¬的点可计算其值。线路的长度等于其边数，一个多项式长度的线路族是一布尔线路的无限序列：$C_1,C_2,...$，使得对每一个 n，电路 C_n 有 n 个输入结点，且长为 $p(n)$，这里 $p(\cdot)$是一多项式（适用于整个线路族）。

输入长度为 n 的图灵机 M 的算法可以用有 n 个输入节点、长度为 $O((|\langle M\rangle|+n+t(n))^2)$的一个线路来模拟，这里 $t(n)$依赖于输入为 n 的算法机 M 的运行时间。这样，一个非均匀多项式时间算法机序列就可以被一个多项式长度的非均匀线路族来模拟。反过来也正确，因为具有多项式描述长度的算法机能够包含多项式长度的布尔线路，并能在多项式时间内模拟其计算。利用线路公式的好处是不需要像第一个公式那样两次（一次是长度，一次是时间）重复多项式要求。

约定　为了简便，我们经常不厌其烦地讨论线路族$\{C_n\}_{n\in\mathbb{N}}$，这里 C_n 有 poly(n)个输入位，而不是 n 个。

1.3.4 难处理假设

我们应该把那些概率多项式时间算法机不能完成的工作认为是难处理的。然而，我们感兴趣的攻击者的工作（"攻破加密算法"、"伪造签名"等）却可以通过非确定多项式时间算法机来进行（因为一旦找到了方法，就很容易验证它的有效性）。这样，只有当 $\mathcal{NP}$ 不包含于$\mathcal{BPP}$ 中时（显然在这种情况下 $\mathcal{P}\neq\mathcal{NP}$）[①]，加密算法（尤其是本书中大部分内容）才显得有意义。这里我们说"无意义"（而不是"无效"），因为我们的每一个陈述都是基于以下形式："如果〈难处理假设〉则〈有用的结果〉"。这样陈述即使在 $\mathcal{P}=\mathcal{NP}$ 或者〈难处理假设〉是错误的情况下仍然有效（〈难处理假设〉一点儿也不比 $\mathcal{P}\neq\mathcal{NP}$ 弱）。但是在这种情况下导出的结果就没有什么意义，因为一切都是从一个错误的前提推出的。

许多地方我们说"如果〈难处理假设〉则〈有用的结果〉"，这些情况下〈有用的结果〉要么意味着〈难处理假设〉，要么意味着比它更弱的形式，这反过来可推出 $\mathcal{NP}\setminus\mathcal{BPP}\neq\emptyset$。因而依据现有的复杂性理论知识，我们不能希望在没有任何难处理假设的前提下确定〈有用的结果〉。

在少数情况下，对概率多项式时间算法机局限性的假设是不够的，所以我们代之以非均匀

① 注意：目前不知道 $\mathcal{NP}$ 是否包含 $\mathcal{BPP}$，故推测 $\mathcal{NP}$ 不包含于 $\mathcal{BPP}$，而不是 $\mathcal{BPP}\neq\mathcal{NP}$。同样，虽然"足够强"的单向函数意味着 $\mathcal{BPP}=\mathcal{P}$，但是，不知道这个等式是否无条件成立。

多项式时间算法机局限性的假设。后者当然要强一些。因为结果是用非均匀复杂性分析的，所以也要强一些。但是，因为我们的证明都是通过简化得到，所以依据概率多项式时间算法得出的结果会比依据非均匀多项式时间算法得出的结果更强一些。除非不为人所知或太复杂，否则人们将更倾向于前者。实际情况确实是这样，因为概率多项式时间算法变形（用概率方法证明隐含结果）通常包含了非均匀多项式时间算法推理（用非均匀方法证明论点），但反过来却不一定正确。[①]

最后，我们指出复杂性最坏（如 $P \neq NP$）条件的难处理假设是不充分的，因为其相应的结果不能令人满意。只保证在最坏的情况下不能被攻破的密码算法是无用的。一个好的密码算法必须是在“大多数情况下”（也就是说，典型情况下）不能被攻破，这意味着在平均意义上该算法难于被攻破。同样，因为我们不能证明“最坏情况下的难处理”是否意味着一般情况下的难处理（如果能证明的话，这将会是复杂性理论的一个重大突破），所以我们的难处理假设必须考虑平均意义上的复杂性。

1.3.5　预言机 (Oracle Machine)

在复杂性理论中使用预言机最初是为了获得可约性概念。本书（主要是第 5、6 章）里我们使用预言机则是出于完全不同的目的——模拟攻击者在试图攻破系统的过程中可能采取的密码系统。预言机的其他一些用途将在 3.6 和 4.7 节中讨论。

简单地说，预言机是能够被扩展向外界提问的一种机器。我们来看一个具体的实例，在这个实例中，始终通过被称做预言的一些函数 $f:\{0,1\}^* \to \{0,1\}^*$ 来回答问题（也叫质疑）。也就是说，如果预言机提出问题 q，那么它得到的答案是 $f(q)$。此时我们说预言机拥有预言 f。

定义 1.3.8（预言机）：所谓的确定性或概率预言机实际上是一种确定性或概率图灵机。这种图灵机有一个被称做预言带（oracle tape）的附加带（additional tape），以及两个独特的状态：一个是预言请求（oracle invocation）状态，另一个是预言作用（oracle appeared）状态。当输入为 x、预言为 $f:\{0,1\}^* \to \{0,1\}^*$ 时，确定性预言机 M 的计算是根据连续配置关系来定义的。对于非预言请求状态的配置，它的下一配置和平常一样定义。设 γ 是请求状态下的一个配置，且预言带上的内容是 q，那么紧接 γ 的另一配置也必是 γ，除非状态是预言工作态，且预言带的内容成了 $f(q)$。称 q 为 M 的质疑，$f(q)$ 为预言回答。概率预言机算法的定义基本类似。当输入为 x、预言为 f 时，预言机 M 的输出分布用 $M^f(x)$ 表示。

这里我们要特别指出：预言机的运行时间是指它的计算步数，而预言机每回答一次问题只需一步。

1.4　严密处理的目的

这一部分我们讨论 3 个相关的问题：

1. 特定领域对严密处理的需要；

---

① 在读过一些具体的例子后，本段会理解得更好。

2. 严密处理的实际意义和/或结果；

3. 对严密处理的传统看法和倾向。

第 2、3 两个问题可能在阅读其他章节后会变得越来越清楚。

1.4.1 严密处理的需要

如果一个命题的真实性不能由
我们已认可的事实推导出
那么这个认可的事实就与命题的真实性无关
Ludwig Wittgenstein，Tractatus logico-philosophicus(1921)

密码学关心的是算法的结构问题。这些结构足以应付恶意的攻击，而这些攻击往往使它们偏离原来设想的功能。给定一个期望达到的效果，密码学家就应该设计一种算法，不仅在"正常工作"时必须能达到效果，而且在算法设计完成后面对将会出现的各种攻击者的攻击，它应该仍然能继续有效(正常工作)。问题是攻击者总是在算法完成之后才进行攻击，从而使得这种算法的设计非常困难，特别是攻击者总是尽力使用不同于设计者预想的攻击方法。因而密码算法的评估必须考虑一个实际上无限大的攻击策略集合，但设想一个攻击者可能使用的特殊策略是无用的，惟一被证明有效的方法是考虑攻击者的计算能力。总的来说，密码算法的评估实际上是对无限大的潜在攻击策略集合的研究，这些攻击策略是不明确的。如此高难度的研究必须投入巨大的精力才能进行(也就是说，要非常严密)。

密码算法系统的设计必须有严格的理论基础，专门的方法和试探性方法都是非常危险的。在设计者对算法运行的环境很清楚时，试探性方法可能会有用，但它总是没有经过严密分析的方法好。而且密码算法必须在恶意选择的环境中运行，这种环境通常是超乎设计者想象的。在这种情况下，试探性方法毫无意义。

除了这些直接的讨论，我们做两点额外的补充。

关于对不可靠直觉的信任 我们认为科学的一个职能就是清楚地解释我们对现实的直觉，并对它进行检验和修正。为了进行检验，必须有一个严密的公式。通过检验，我们的直觉可被证实或证伪(或被判定为只在特定情况下或经过一定修正后才成立)。许多情况下我们的直觉是正确的，也有许多情况我们的直觉是错误的，我们理解规律越深刻，直觉就会变得越正确。

在目前的历史阶段，断言我们对"高效计算的本质"有很好的直觉是毫无根据的。特别是，我们甚至不知道一些基本问题的答案，比如 $\boldsymbol{P}$ 是否严格包含于 $\boldsymbol{NP}$，更不用说理解是什么使一个计算问题变得复杂，而使另一个表面看来相关的问题变得简单。因而我们在判断哪些问题可被高效计算，哪些问题不能时，应该非常慎重。遗憾的是，这种判断恰恰是密码学要研究的内容。更糟的是，许多密码学问题有着比通常所见的复杂性理论更复杂和更难懂的描述。总之，密码学处理的是一些非常复杂的计算问题，目前必须在对非常简单的计算概念没有很好理解的情况下来进行。因此，目前我们对密码学的直觉认识在没有得到清楚解释和认真检测之前，只能认为是非常不可靠的。换句话说，在像密码学这样一个与我们难以了解的问题紧密相关的敏感领域，对直觉的解释和对检测的需求变得越来越强烈。

跟踪记录 密码学是一门振奋人心的学科，因此许多科研人员、工程师甚至外行人员都在

讨论密码学问题。遗憾的是，大多数讨论都是在对研究对象没有一个精确定义的情况下进行的。相反，他们认为密码学的基本概念（如安全加密）是不言自明的，因而也没必要提供详尽的定义。这种假设的错误充分体现在许多在安全方面得出错误结论的论文中（不包括私人讨论），大多数情况下，这种错误结论都可以追溯到一个安全方面的隐含的错误概念。如果这样的概念清楚表达出来，作者就不会犯这样的错误。在这里我们不准备对这类问题一一叙述，但是有一个著名的例子是必须提的。

大约在1979年，Ron Rivest声称"在因数分解是难处理的假设下被证明是安全"的签名算法中，没有一个能抵抗"选择明文攻击"。他的论断建立在对"安全认证"实质进行假设的基础上（假设因数分解是难处理的），这种假设既没有明确表述也没有得到证实。因此，以后几年中人们都徘徊在两种选择中间，到底是研究在因数分解是难处理的假设之下被证明是不可伪造的签名算法，还是研究能抵抗"选择明文攻击"的签名算法。然而，在1984年Goldwasser，Micali和Rivest指出1979年Rivest的论断基础错误，并且找到了在通常假设下能抵抗"选择明文攻击"的签名算法。特别地，因数分解是难处理的足以证明存在一种即使在"选择明文攻击"的情况下也能抵抗"伪造"的签名算法。

总之，密码学的基础概念确实很普通，但却不是不证自明的，而且还是不好理解的。我们目前对这些概念的理解还很不够，如果不采用准确的定义并对每项陈述都做严密论证的话，就不能进行正确地讨论这些概念。

1.4.2 严密处理的实际结果

按照复杂性理论的常规做法，我们的处理方法以算法的渐近分析的方式提出（实际上采用"运行时间函数解析"的说法会更准确一些）。这使我们的处理少了许多麻烦，但对基本思想却不是不可或缺的。特别地，本书所使用的一些定义方法（例如：单向函数、伪随机发生器、零知识证明、安全加密算法、不可伪造签名算法、安全协议等的定义）都基于通用格式，这些格式在任何合理的计算模型中都是有效的。特别地，尽管这些定义都以"抽象的形式"给出，却都留了具体的添加补充余地。一些通常与几个这样的定义相关的结果也具有同样的特性。为了阐明前述观点，也作为一个例子，下面我们看一下本书里提到的一个普通的结果。

本书中一个有代表性的结果与两个计算问题相联系。第一个问题是假设为难处理的一个简单计算问题（如因数分解的难处理性），第二个问题由对特定的密码学算法（如一个特定的加密算法）的"破译"方法组成。抽象定义可能断言，如果多项式时间算法不能处理整数分解问题，则这个加密算法是安全的，即它不能在多项式时间内被攻破。我们可以证明：如果整数能在多项式时间内分解，则加密算法可被攻破。因此我们实际上正在证明的是：如果某人在时间$T(n)$（这里n是安全参数，例如密匙长度）内能破译算法，那么他一定能在时间$T'(m)=f(m, T(g(m)))$内将长度为m的整数分解因子，这里f和g是证明过程中两个固定的多项式。要想评价一个结论的实际意义，首先应考虑这些多项式（f和g）。本书推出的大多数基本结论中这些多项式是相当小的，拥有合理的安全参数和合理的难度假设的算法总是可以成为实际不可破译的算法。在一些异常情况下，我们清楚地阐述并把这些结果仅仅看成是对联系两个概念的合理性的声明。实际上我们把结论分成三类：

1. 似乎可行的结论：这里，我们是想建立两种概念之间的联系，或者致力于提供解决一类

问题的方法。

第一类结论认为，从原则上讲，为构造 Y(如一种有用实施)可使用 X(如一种特定工具)，然而在论证过程中提出的具体结构却可能是不切实际的。虽然如此，这样的结论仍可能是有用的，因为它暗示如果想找到构造 Y 的方法，可以运用 X 的特殊实现。最起码这样的结论可以给研究人员提出一个挑战：要么提供一个用 X 构造 Y 的实际办法，要么解释为什么这种方法不可行。

第二类结论认为，任何一个问题只要是属于 C 类的就是可解的，但论证提出的一般构造方法却可能是不切实际的。然而这仍是一条很有价值的信息：如果遇到属于 C 类的特定问题，则可推出它在理论上是可解的。如果需要构造一个真正的系统，可能应该设计新方案(而不是应用前述的一般结论)。

总之，在这两种情况下，似乎可行的结论都给我们提供了有用的信息(即使它不能提出实际的解决方案)。此外经常会发现一些用来证明结果可行性的工具常常会对解决手边的特定问题有帮助，这正是下一类结论所要阐述的。

2. *在实际中可能使用的范例和技术的介绍*：这里我们指那些旨在推出新概念、新模型、新工具和新技术的理论成果。这些理论成果(例如：实用技术)通常具有实用性，不管是最初提出的原创性理论还是经过修正，均是如此，至少可以起一种启发的作用。

3. *适用于实践应用的算法的陈述*。

一般很容易决定一个特定的理论成果属于前述的哪一类，遗憾的是原始论文并不都进行归类，但一般情况下可以从结构中明显看出所属的类。特别指出：目前所知的所有理论成果(包含本书中我们讨论的所有理论成果)都有一个明确清晰的结构。更进一步讲，理论成果的结构的安全性与特定难处理任务的复杂性是有明确关系的。与一些不成熟的想法相反，对每个这样的理论成果，存在一个明确的换算，即从具体的难处理假设(算法的基础)到要破坏结论算法安全性所要的工作量下界的换算。[①] 特别指出：这些换算可作为安全参数的任意值。这样做便可知道在特定的合理的难处理假设下对特定应用来说一种特定的构造是否足够。许多情况下回答是肯定的。但一般来说，这个答案确实依赖于特定的构造、安全参数的特定值和对这个值(安全参数)估计的合理性。

1.4.3 保守倾向

当读到对密码学基本概念进行定义的章节时，读者可能会注意到在对安全定义时，我们不现实地“保守”。换句话说，对安全的定义应该是谨慎的。从技术上说，这种定义倾向不会产生任何问题，因为强安全概念从一个更合理即有更多限定的意义上看，当然也是安全的。此外，还能用合理的难处理假设来实现(强安全)概念。大多数情况下会发现这些假设即便对更弱的安全概念也是必需的。现在读者可能会想知道我们为什么选择看起来比实际要求更强的定义。

之所以在安全定义时有保守倾向，其原因是：要想准确知道在特定的实际应用中对安全的要求实在太难了。此外，每一个实际应用都有不同的要求，而且为每一个新应用重新设计系统

① 后一论断的惟一例外是 Levin 关于通用单向函数存在性的观察(见 2.4.1 节)。

也是不方便的。因此，我们实际上需要考虑所有未来（目前不知道）的安全方面的问题，而不仅仅是关注一些已知的安全方面的应用。不用保守方法，要想包括所有应用方面的要求似乎是不可能的 。[①] 后面我们将会看到保守方法如何对安全进行定义，以便包含所有可能的实际应用中的安全问题。

1.5 其　他

在图 1.1 中，我们用图中文件提出的方法对密码学进行"历史回顾"（也就是 20 世纪 70 年代中期的情况）。

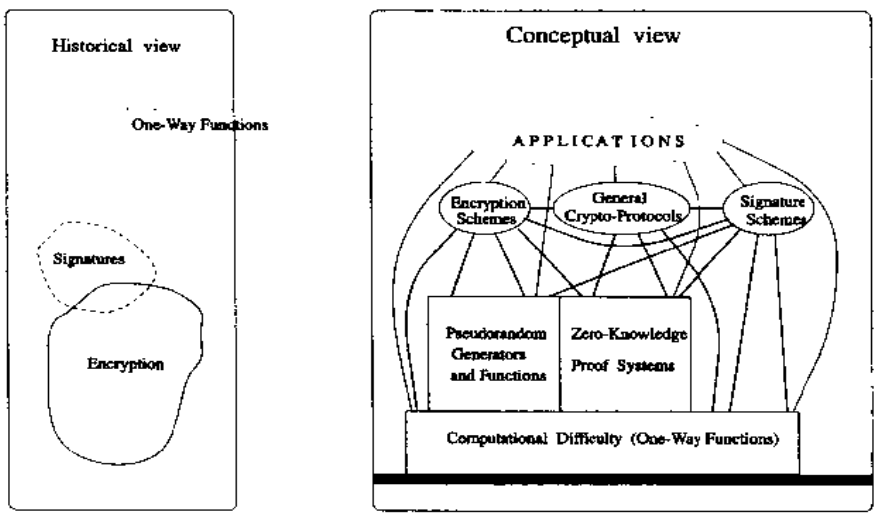


图 1.1　密码学：两种观点

1.5.1　历史记录

我们主要来看一下 20 世纪 80 年代所做的工作，它们在今后的论述中起着重要的作用。这些工作同样受到了前期工作的巨大影响，但在后面的章节中我们没有明确指出前期工作的这些影响。这一节里我们要讨论前期工作的一部分影响。一般来讲，这些影响包括建立直觉目标、提供基本手段、提出一些可能的解决方案，所有这些构成了构造性评判的基础，从而得到更健壮的方法。

古典密码学　古典密码学对密码学中根本性问题的回答是很悲观的（也就是说，不可能设计出一种不可被攻破的密码）。Shannon[200]甚至建议对问题本身进行修正，即我们应该问破译一种密码是否可行，而不是是否可能。一种代码如果攻击者破译所需的工作量与合法用户译码时的工作量成合理比例的情况下，依然不能被攻击者破译，则被认为是好的。实际上，这正是现代密码学所采用的方法。

---

① 有人会说，如果不采用我们的保守方法，连包括一种应用方面的要求看起来也是不可能的。

密码学的新方向 商业应用的前景触发了加密算法用于民用方面的投资。20 世纪 70 年代初设计的 DES 分组码[168]采用了新的模式(paradigm),显然要想破译它可能但不可行。伴随构造和分析新加密算法(私人密匙)的挑战又有了新的问题,像如何在不安全信道交换密匙等等[159]。同时还出现了新的概念,如数字签名(见 Diffie 和 Hellman[63]和 Rabin[186])、公匙密码系统[63]和单向函数[63]。Merkle 和 Hellman[163]、Rivest、Shamir 和 Adleman[191]以及 Rabin[187]等人最早建议使用这些概念。

20 世纪 70 年代后期密码学与复杂性理论直接联系起来[39,73,147]:有关破译 1-1 密码图表的问题不是 ***NP*** 完全问题,更重要的是,破译任务中存在的 ***NP*** 难题为密码安全所能提供的支持非常有限。后面将介绍"$2n$ 中出 n 的认证"[186]和秘密共享[196]技术,这些问题将在后续的研究中广泛应用。

新世纪的曙光 对密码协议的早期研究暴露了安全方面一些含糊概念的缺陷,也显示出了密码协议设计的精妙所在。特别地,电话抛币[31]、秘密交换[30]、健忘传输[188]([70])这样的问题都得以明确表述。对[199]中提出的智力扑克协议安全性的怀疑(由 Lipton 提出)引出了安全加密的概念,也引出了诸如计算不可分辨性这样的概念[123,210]。由 Fischer 提出的对健忘传输协议[188]的质疑,引出了由 Goldwasser、Micali 和 Rackoff[124]早在 1982 年 3 月提出的零知识证明的概念。

在[65]中提出了安全密码协议的规范方法。这个方法实际上标识了一个不安全协议(即可被句法限定类型攻击攻破的协议)的子类。此外还指出对于一个特定协议,很难检测它的安全性[69]。回想一下,我们现在的任务是要组建一个安全协议,并提供其安全性的证明,并且这个方法是完备的(即:用它能解决一切可以解决的问题)。

1.5.2 关于进一步阅读的建议

在 1.2 和 1.3 节里我们分别给出了概率论和计算复杂性理论的背景材料,这是实现我们这本书的目的所必需的。如果读者对这方面的知识不太熟悉的话,最好看一些概率论(如[79])和算法复杂性理论(如[86,202])的比较好的课本,同样熟悉一些随机化计算也是大有裨益的[167,97]。

数论中的计算问题为单向函数的广泛应用提供了保障(与难处理的假设联系起来)。然而本书着重强调概念,而不是这些概念的具体应用,这些内容(数论)将不会是本书的主要内容。因此计算数论的背景对本书不是非常必要,但我们还是在附录 A 中对相关问题做了一些简单陈述,有兴趣的读者可参阅其他课本(如[10])。

本书重视基本概念、定义和密码学方面的一些理论成果。如前所述,这些对完美的实践是至关重要的。然而只有这些正确的理论体系对实践来说还是远远不够的,但本书不会提供理论体系之外的任何东西。要想获得实践方面的建议(如应用密码学),读者可借鉴地参考别的课本(如[158])。

我们用对算法的渐近分析处理问题。简言之,我们要的是能很好抵制多项式时间算法的攻击,而不是针对一般的时限攻击。然而正如 1.4 节解释的那样,所有这些陈述都不是实质性的。和 2.6 节或其他任何地方一样,结构都以通常的方式陈述(如[155])。我们选择耐抵抗多项式时间算法攻击,使对结果的陈述稍微简单一些,因为这样避免了安全认证过程中的许多复

杂变形。这对阐述一些可行的理论成果已足够，也正是本书的目的所在。但当算法付诸实践时，还必须对其变形的复杂性相当了解(因为这将决定一个具体算法的实际安全性)。一般情况下本书中出现的变形复杂性很好掌握，某些情况下我们也会提到一些有关这方面的结论。这里要另外指出的是：Luby 的书中提到的是针对一般时限攻击的理论成果。

1.5.3 未决问题

如前所述(下一章将做进一步阐述)，本书的大多数内容都依赖于单向函数的存在。目前，我们假定这类函数存在，希望在新世纪中能证实这个得到普遍接受的假定。单向函数的存在就意味着 **NP** 不包含于 **BPP**，因此也就说明 **NP**≠**P**(这是计算机科学领域最著名的未决问题)。

我们不知道 **NP**≠**P** 是否隐含一些实际结果。以后，需认为许多难的实例不仅存在，而且要频繁出现(如一些简单抽样分布)。我们将在第 2 章做进一步讨论。

1.5.4 习题

习题 1：马尔可夫不等式应用：

1. X 是一随机变量，使得 $\mathsf{E}(X)=\mu$ 且 $X\leq 2\mu$，试给出 $\Pr\left[X\leq\frac{\mu}{2}\right]$的一个上界。

2. 设 $0<\epsilon$ 且 $\delta<1$，随机变量 Y 在$[0,1]$之间取值，使得 $\mathsf{E}(Y)=\delta+\epsilon$，试给出 $\Pr[Y\geq\delta+\frac{\epsilon}{2}]$的一个下界。

提示：上面两题中，自己可构造一辅助随机变量，再应用马尔可夫不等式，但直接运用证明马尔可夫不等式中的推理方法会更简单一些。

习题 2：Chernoff/Hoefding 界的应用：设 $f:\{0,1\}^*\to[0,1]$是一个多项式时间算法计算函数，记 f 在$\{0,1\}^n$ 上的平均值为 $F(n)$，即

$$F(n)\stackrel{\text{def}}{=}\frac{\sum_{x\in\{0,1\}^n}f(x)}{2^n}$$

设 $P(\cdot)$为一个多项式函数，试给定一种概率多项式时间算法，当输入 1^n 时输出约为 $F(n)$，记为 $A(n)$，使得

$$\Pr\left[\,|F(n)-A(n)|>\frac{1}{p(n)}\right]<2^{-n}$$

提示：算法以随机多项式选取许多(一共有多少?)个采样点 $S_i\in\{0,1\}^n$，这些采样点相互独立，并服从正态分布(为什么?)。算法输出这些采样的平均值，利用 H 不等式分析算法的运行过程(提示：定义随机变量 X_i 使得 $X_i=f(s_i)$)。

习题 3：对图联结性算法的分析：考虑 1.3.2.1 中给出的算法，表明：如果图 G 中 s 与 t 相连，那么以 s 为起始点的任意路径至少有$\frac{2}{3}$的概率经过结点 t。

提示：考虑与结点 s 的连接分量，设为 $G'=(V',E')$。对 E'中的任一边(u,v)，使 $T_{u,v}$为一随机变量，表示从 u 出发沿任意路径最终能够到达 v 的步数。首先证明 $\mathsf{E}[T_{u,v}]\leq 2|E'|$(提示：考虑从特定方向在随机行中经过这条边的“频率数”，注意这个频率是不依赖于边和方向

的)。其次记 cover(G')为起点为 s、终点为 V' 的最后一个定点的这种路径在一个随机路径中的期望长度,证明:cover(G')$\leq 4\cdot|V'|\cdot|E'|$(提示:考虑经过 G' 的所有顶点的一个环路,可以看出(G')$\leq\sum_{(u,v)\in c}\mathsf{E}[T_{u,v}]$)。引用马尔可夫不等式可得出结论。

习题 4:*BPP* 的另一定义。 Part1:证明对于每一个正多项式 $p(\cdot)$,当用$\frac{1}{2}+\frac{1}{p(|x|)}$代替$\frac{2}{3}$时,定义 1.3.4 将更有说服力。即,如果存在一个多项式 $p(\cdot)$ 和一个概率多项式时间算法机 M,满足:

- 对所有 $x\in L$,有 $\Pr[M(x)=1]\geq\frac{1}{2}+\frac{1}{p(|x|)}$
- 对所有 $x\notin L$,有 $\Pr[M(x)=0]\geq\frac{1}{2}+\frac{1}{p(|x|)}$

则有 $L\in$ ***BPP***。

提示:给定一个满足上述条件的概率多项式机 M,如下构造概率多项式算法机 M'。对于输入 x,M' 执行 M 共 $O(p(|x|)^2)$ 次,每次以 x 为输入且遵循多数决定的原则。利用 Chebyshev 不等式(见 1.2 节)说明 M' 至少有$\frac{2}{3}$的概率是正确的。

习题 5:*BPP* 的另一定义。 Part2:证明对于每一个正多项式 $p(\cdot)$,当 $1-2^{-p(|x|)}$ 代替$\frac{2}{3}$时,定义 1.3.4 将更有说服力。也即,对所有 $L\in$ ***BPP*** 和每一个多项式 $p(\cdot)$,存在一概率多项式时间算法机 M,使得

- 对所有 $x\in L$,有 $\Pr[M(x)=1]\geq 1-2^{-p(|x|)}$
- 对所有 $x\notin L$,有 $\Pr[M(x)=0]\geq 1-2^{-p(|x|)}$

提示:除了需要使用更强的概率不等式(即 Chernoff 界,见 1.2 节)外,其他类似于习题 4。

第2章 计算复杂性

在这一章中我们定义并学习单向函数。单向函数提出一个"有用"的计算复杂度的概念，也是本书大多数结论的基础。简单地说，单向函数是一个容易计算函数值而不容易求逆的函数(在通常意义下来说)(看图 2.1 的图解)。特别地，我们定义了强的和弱的单向函数，并证明了弱单向函数的存在隐含了强单向函数的存在。证明给出了可约性理论(reducibility argument)的一个很好的例子，实际上也是建立这个领域大多数结论的"可约理论"的很严格的例子。此外，这个证明还提供了一个简单的实例，即计算问题比"信息-理论模拟"更难证明的例子。

此外我们定义核心断言(hard-core predicate)，并证明每个单向函数都有一个核心断言。在今后几乎所有章节中，核心断言都起着重要作用(签名算法例外)。

结构安排 在 2.1 节中，我们简单说明了在不同的密码学原始概念中都隐含了单向函数的概念，从而使单向函数的定义成为必然。2.2 节给出了基本定义，2.3 节指出可用弱单向函数构造强单向函数，2.6 节提出了一个更有效的构造方法(在特定的严格条件下)。在 2.4 节中，我们将单向函数看作有限函数的一致集合，并讨论了这些集合可能有的各种附加性质。2.5 节定义了核心断言并指出如何由单向函数来构造它们。

教学提示 如前所述，弱单向函数的存在隐含强单向函数的存在(见 2.3 节)的证明对其余的理论都具有启发性。因此，如果略去了这个证明，在以后首次使用可约性理论时一定要插入对它的讨论(例如，在指出如何由单向函数构造核心断言时)。

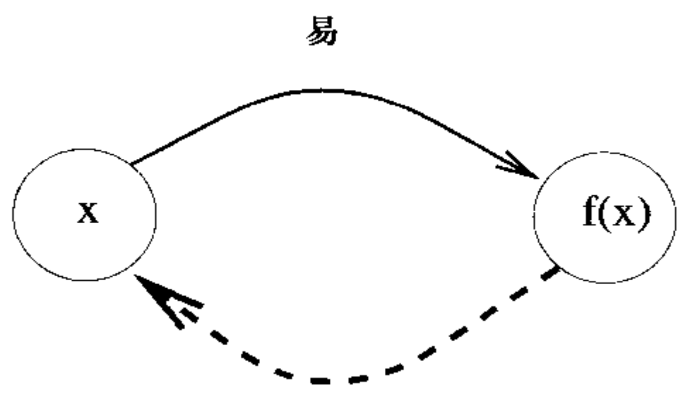


图 2.1 单向函数的示意图

2.1 单向函数:动机(单向函数的意义)

如前言所述,为合法用户提供有效算法,并使非法攻击行为篡改和破译这些算法的计算不可行,现代密码学就是建立在这样一个对抗的基础上的。我们以安全数字通信中的密码学手段,也即加密算法为例来说明这种对抗。

在安全的加密算法中,合法用户应该能利用他们的某些私人信息很容易地从密文恢复明文,而攻击者(没有这些私人信息)却不能有效地对密文解密(也就是说,在概率多项式时间内)[①]。另一方面,一个非确定机却能很快地将密文解密(如:通过猜测私人信息)。因此,安全加密算法的存在就意味着有这样一种工作(如:"破译"加密算法),这种工作只能由非确定多项式时间机而不能由确定多项式时间机(即使是随机的)来完成。换句话说,安全加密算法存在的一个必要条件是 NP 不被 BPP 包含(从而:$P \neq NP$)。

尽管 $P \neq NP$ 是现代密码学的一个必要条件,但却不是一个充分条件。假设破译一个加密算法是 NP 完全的,那么 $P \neq NP$ 意味着这种加密算法在最坏的情况下也是难以攻破的,但它仍然不能排除一个加密算法在绝大多数情况下很容易被攻破的可能性。实际上,可以构造一个破译问题为 NP 完全,但同时存在一个能以 99% 概率被成功破译的一个加密算法。因此,在最坏的情况下难破译不是安全性的一个好的评估。安全要求在绝大多数情况下难破译,或至少"在通常情况下是难破译的"。安全加密算法存在的一个必要条件是通常情况下难计算的 NP 问题的存在。必须指出:现在还不知道 $P \neq NP$ 是否隐含了通常情况下难计算的 NP 问题的存在性。

此外,只考虑通常情况下难计算的 NP 问题的存在性是不够的。为了能应用在通常情况下难计算的问题,我们必须能够找到一些难计算的实例和能很快解决这些难题的辅助信息。否则,它们对合法用户也是难处理的,这样,合法用户相对于攻击者就没有计算优势了。因此,安全加密算法的存在就意味着存在生成带有相应辅助输入的实例的有效方法(也就是概率多项式时间算法),使得:

1. 给了辅助输入,很容易解决这些实例;
2. 通常情况下,如果不给出这些辅助输入,解决这些实例很困难。

前述要求反映在单向函数的定义中(下一节介绍)。简单地说,单向函数是易于计算函数值而难于求逆(一般情况下)的函数。这样,单向函数抓住了生成实例过程的逆过程的困难性(并从实例本身获得附加输入),这也正是造成前面两个条件之间差异的原因(习题 1 将进一步讨论这个关系)。

假设单向函数存在,也就是假设这样的函数,其存在有效的正过程(也就是函数的正向计算),而求逆运算很难。注意,在日常生活中这样的过程很多(如火柴的燃烧)。这样,假设存在单向函数,就成了日常生活中复杂理论的模型。

---

① 这里的"私人信息"称为密钥,见第 5 章。

2.2 单向函数的定义

在本节中，我们给出单向函数的几种定义。第一种定义我们今后称之为强单向函数(或单向函数)，是最普通的一种定义。此外还给出了弱单向函数、非均匀单向函数的定义，以及这些函数可行的候选形式。

2.2.1 强单向函数

简单地说，单向函数是一种易于计算而难求逆的函数。前一个条件很明显：函数 f 易于计算意味着存在一个多项式时间算法，使得当输入为 x 时输出为 $f(x)$。第二个条件就必须好好推敲了，我们说函数 f 难求逆是指，给定一个输入 y，要求在 f 作用下 y 的逆元，通过概率多项式时间算法来试探，能够成功的概率很小很小，几乎可以忽略(对 $|y|$)，这里成功的概率与 y 的选择有关(将在后面讨论)。如果对每一个正多项式 $p(\cdot)$ 和足够大的 n，都有 $S_n<\frac{1}{p(n)}$ ($\mu(n)<\frac{1}{p(n)}$，则称序列 $\{S_n\}_{n\in\mathbb{N}}$(函数 $\mu:\mathbb{N}\to\mathbb{R}$)对 n 是可以忽略的。下面是进一步的讨论。

定义 2.2.1(强单向函数)：函数 $f:\{0,1\}^*\to\{0,1\}^*$ 如果满足下面两个条件：

1. 易于计算：存在一个(确定的)多项式时间算法 A，使得当输入为 x 时算法 A 的输出为 $f(x)$(即 $A(x)=f(x)$)。

2. 难于求逆：对于每一个概率多项式时间算法 A'、每一个正多项式 $p(\cdot)$ 和所有足够大的 n，都有：

$$\Pr[A'(f(U_n),1^n)\in f^{-1}(f(U_n))]<\frac{1}{p(n)}$$

则称函数 f 为(强)单向函数。

回想一下，U_n 是服从 $\{0,1\}^n$ 上的均匀分布的一个随机变量，因此条件 2 中的概率要由服从均匀概率分布的 U_n 的所有可能取值以及所有可能的内部抛币作为 A' 的输入而得到。注意并不要求 A' 输出 $f(x)$ 的特定原像，任意原像(即相当于 $f^{-1}(f(x))$ 的元素)都可以(实际上，如果 f 是 1-1 的，则串 x 是在 f 作用下 $f(x)$ 的惟一原象，但在一般情况下，$f(x)$ 还可能有其他的原像)。

辅助输入 1^n　对于求逆算法 A'，不仅要给出函数 f 值域中的一个取值，还要给出要求输出的长度(用一元记号表示)。这种约定的主要原因是排除人们误解单向函数概念的可能性，即单向函数之所以为单向函数，是因为它激烈地压缩它的输入，从而使得求逆运算没有足够时间打印想要的输出(即相应的原像)这种错误的概念。例如：我们定义函数 $f_{len}(x)=y$，使得 y 是 x 长度的二进制表示(即 $f_{len}(x)=|x|$)。因为 $|f_{len}(x)|=\log_2|x|$，所以当输入 y 时，没有一种算法能够在 $|y|$ 的多项式时间内对 f_{len} 求逆得到 y 的原像，但却存在能够在 $|x|$ 的多项式时间内当 $y=f_{len}(x)$ 时求 f_{len} 的原像的算法。(例如，通过 $|x|\mapsto 0^{|x|}$)。一般来说，如果辅助输入 $1^{|x|}$ 与输入 $f(x)$ 相结合，就可以得到一个运行为主要输入和要求输出的总长度的多项式时间的求逆算法。注意到在特定的等长度的函数 f(即对所有的 x 都有 $|f(x)|=|x|$)算法下，辅助输入是多余的。更一般地，如果只给出 $f(x)$，就可以在 $|x|$ 的多项式时间内计算 $1^{|x|}$

(见习题 4 和 2.2.3.2 节),则辅助输入也是多余的。

进一步讨论

前面定义中的求逆难度可以看作高效求逆算法成功概率的一个上界,这个概率由求逆算法的随机选择和选择算法的(主要)输入(即 $f(x)$)的分布来决定。将 f 作用于 $x\in\{0,1\}^n$ 上均匀选择的 x,就可以得到求逆算法的输入分布。如果 f 导出$\{0,1\}^n$上的一个置换,则求逆算法的输入也服从$\{0,1\}^n$上的均匀分布。然而,通常情况下,f 不必是一个 1-1 函数,求逆算法的输入分布也就远远不是均匀的了。在任何情况下,定义在前述概率空间上的成功概率按要求都是可以被忽略的(作为长度 x 的一个函数)。为了进一步阐明成功概率的条件,我们考虑两个不证自明的算法。

随机猜测算法 A_1 当输入$(y,1^n)$时,A_1均匀地选择并输出长为 n 的串。需要注意的是,A_1的成功概率等于随机变量 $f(U_n)$的冲突概率(即$\sum_y\Pr[f(U_n)=y]^2$)。也就是说,令 U'_n 表示与 U_n 相互独立且服从$\{0,1\}^n$上均匀分布的随机变量,我们有

$$\Pr[A_1(f(U_n),1^n)\in f^{-1}(f(U_n))]=\Pr[f(U'_n)=f(U_n)]$$
$$=\sum_y\Pr[f(U_n)=y]^2\geq 2^{-n}$$

这里不等号成立利用了这一结论:对于和为 1 的非负数 x_i,当每一个 x_i都相等时,和$\sum_i x_i^2$ 有最小值。这样,最后一个不等式当且仅当 f 是 1-1 的函数时等号成立。从而:

1. 对每一个函数 f,不证自明的算法 A_1 的成功概率是严格为正的。所以,我们不能苛求所有高效的算法都不能对 f 求逆。

2. 对每一个 1-1 函数 f,A_1 对 f 求逆的成功概率都是可以忽略的。当然,这并不能表明 f 是单向的(尽管 A_1 是不证自明的)。

3. 如果 f 是单向的,则随机变量 $f(U_n)$的冲突概率是可以忽略的。这是因为 A_1 属于定义的范围,它的成功概率等于冲突概率。

固定输出算法 A_2 另一种通俗的算法记为 A_2,这种算法用于计算对相同长度的所有输入有相同函数值的函数(例如,$A_2(y,1^n)=0^n$)。对每一个函数 f,有

$$\Pr[A_2(f(U_n),1^n)\in f^{-1}(f(U_n))]=\Pr[f(0^n)=f(U_n)]$$
$$=\frac{|f^{-1}(f(0^n))|}{2^n}\geq 2^{-n}$$

当在 $f(0^n)$作用下有惟一原像(即 0^n 自己)时,上面等号成立。我们再考虑一些类似的事实:

1. 对任意函数 f,通俗的算法 A_2 成功的概率是严格为正的。

2. 对任意 1-1 的函数 f,A_2 对 f 求逆成功的概率是可以忽略的。

3. 如果 f 是单向的,那么在$\{0,1\}^n$中,被 f 映射为 $f(0^n)$的那部分 x 是可以忽略的。

很明显,定义 2.2.1 考虑的是所有的概率多项式时间算法,而不仅仅是前面讨论的通俗的算法。从某种意义上说,这个定义告诉我们,对单向函数来说,没有一个概率多项式时间算法能够比这些通俗的算法更好。

可忽略概率

我们给出几个有关可忽略概率的术语。前面的定义和讨论认为,如果作为输入长度的一个函数,其成功概率以每个多项式分数为上界,则这个算法的成功概率可以被忽略。从而,重

复算法多项式(输入长度的)许多次,便可以得到一个新的算法,它也有可忽略的成功概率。换句话说,以可忽略概率出现的事件,即使在重复(n 的)多项式次的试验中,出现的概率也是可以忽略的。因此,在定义可行性计算为"可以在多项式时间内被计算"时,可以同样地定义可忽略成功率为"发生的概率比任何多项式分数都小"。

"可忽略分数/概率"概念的一个反面是"显著的分数/概率"。如果存在一个多项式 $p(\cdot)$,使得对于所有足够大的 n,有 $\mu(n)>\frac{1}{p(n)}$,我们就称函数 $\upsilon:\mathbb{N}\rightarrow\mathbb{R}$ 是显著的。这里要指出,存在一些函数既不是可忽略的,也不是显著的。

2.2.2 弱单向函数

前面定义的单向函数是很严格意义下的单向,即在求逆的时候任意高效的求逆算法,其成功概率都是可忽略的。下面提供一种比较弱一些的定义,它只要求所有的高效求逆算法以显著的概率失败就可以了。

定义 2.2.2(弱单向函数):如果函数 $f:\{0,1\}^*\rightarrow\{0,1\}^*$ 满足下面两个条件:

1. 容易计算:和强单向函数定义中的要求一样。

2. 稍微有点难求逆:存在一个多项式 $p(\cdot)$,使得对于每一个概率多项式时间算法 A' 和任意足够大的 n,都有

$$\Pr[A'(f(U_n),1^n)\notin f^{-1}(f(U_n))]>\frac{1}{p(n)}$$

则称函数 f 是弱单向函数。

请读者注意下面量词的顺序:存在一个多项式 $p(\cdot)$,使得在 f 作用下求 $f(U_n)$ 原像的所有概率多项式时间算法的失败概率下界都为 $1/p(n)$。

弱单向函数并没有提供前面讨论中暗示的难度类型。不过后面我们会看到弱单向函数能够转化成强单向函数,而后者能够提供难度类型。

2.2.3 两个有用的长度协议

在后面,我们会经常应用下面的两个关于单向函数原像和像的长度协议。本节证明使用这些协议的合理性。

2.2.3.1 为一些长度定义的函数

很多情况下,考虑定义域属于全体字符串集合的单向函数更为方便。特别地,这为介绍一些函数的结构提供了方便。本节后面一直使用的一个特别重要的例子是定义在 $\bigcup_{n\in\mathbb{N}}\{0,1\}^{l(n)}$ 上的函数,这里 $l(\cdot)$ 是某一多项式。对这种实例我们将做更一般的处理。

令 $I\subseteq\mathbb{N}$,关于 I,我们用 $s_I(n)$ 表示 n 的一个后继,即 $s_I(n)$ 是在集合 I 中比 n 大的最小的整数(即 $s_I(n)\stackrel{\text{def}}{=}\min\{i\in I:i>n\}$)。如果存在一种算法,当输入为 n 时,在 poly(n)步内终止并输出 $1^{s_I(n)}$,则称集合 $I\subseteq\mathbb{N}$ 为多项式时间可数的(一元输出使得 s_I 以多项式为界,即 $s_I(n)\leq$ poly(n))。设 I 是一个多项式时间可列集,f 为定义在 $\bigcup_{n\in I}\{0,1\}^n$ 上的一个函数。我们说 f

是定义在长度 I 上的强(弱)单向函数,如果 f 是多项式时间可计算的并且在 I 内对 n 是难求逆的。例如,定义在长度 I 上的强单向函数的难计算条件如下:

对于每一个概率多项式时间算法 A'、每一个正多项式 $p(\cdot)$ 以及 I 内全体足够大的 n,都有

$$\Pr[A'(f(U_n),1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}$$

在前面章节中定义的一般单向函数可以看作是定义在长度ℕ上的单向函数。

在任意多项式时间可列集内的长度上的单向函数可以很容易地转化为一般单向函数(即定义在所有 $\{0,1\}^*$ 上的函数)。特别地,对每一个定义在 $\bigcup_{n\in I}\{0,1\}^n$ 上的函数 f,令

$$g(x) \stackrel{\text{def}}{=} f(x') \tag{2.1}$$

这里 x' 是长度在 I 中的 x 最长的前缀,可以构造函数 $g:\{0,1\}^* \to \{0,1\}^*$。如果函数 f 是等长度的(即对所有的 x 都有 $|f(x)|=|x|$),那么我们就可以修改构造过程以保持这种性质,从而获得等长度的函数 $g':\{0,1\}^* \to \{0,1\}^*$,使得

$$g'(x) \stackrel{\text{def}}{=} f(x')x'' \tag{2.2}$$

这里 $x=x'x''$,且 x' 是长度在 I 中的 x 的最长前缀。

命题 2.2.3:设 I 为一多项式时间可列集,且 f 是定义在长度在 I 内的强(弱)单向函数,那么 g 和 g'(在 2.1 式和 2.2 式中分别定义)在通常意义下是强(弱)单向函数。

尽管上述命题的正确性是非常吸引人的,但我们还是希望读者不要忽略下面的证明。证明过程实际上很简单,第一次用到了后面广泛使用的证明方法。这个结论用反证法来证明,通过假设 $g(g')$ 能够以不可能的概率成功地求逆,得出矛盾,从而证明 $g(g')$ 具有难求逆的性质。得出矛盾的原因在于假设能以不允许的概率成功地对 f 求逆。换句话说,对 f 的求逆被"转化为"对 $g(g')$ 的求逆。这里的"变形"实际上是比标准变形更严格意义下的一种变形,必须保持算法的成功概率不变。这种论点被称为可约性理论。

证明:首先证明 g 和 g' 能以多项式时间算法计算。为了证明这一点,我们利用 I 是一多项式时间可列集的事实,这就意味着我们可以在多项式时间内决定 I 的成员(例如,通过观察 $m\in I \Leftrightarrow s_I(m-1)=m$)。从而当输入 x 时,在多项式时间内可以找到 I 内小于 $|x|$ 的最大的 m。计算 $g(x)$,找到这个 m,并应用函数 f 作用于 x 的前缀的第 m 位。对 g' 可做同样的处理。

下一步我们证明 g 保持了 f 的难求逆的性质,同样的证明可以说明 g' 的难求逆性。为了简单起见,这里只提供 f 是强单向函数时的证明,对于 f 是弱单向函数的情形可类似地给出证明。

用反证法来进行证明。假设命题的结论不正确,即存在一个高效的能以不可忽略的成功概率对 g 求逆的算法。我们用这个算法来构造一个以不可忽略概率对 f 求逆的算法,这样得出一个(与命题前提矛盾的)结论。换句话说,我们指出(以不允许的成功概率)对 f 求逆可以化简为(以不允许的成功概率)对 g 求逆,因此可推导出后一算法的不可能性。化简的依据是如果可以对 g 求任意长度的原像,就可以对 g 求长度在 I 内的原像,而求这样长度的原像 g 与 f 就发生了冲突。

直观地,任意一个对 g 求逆的算法都可以如下转化为对 f 求逆的算法。当输入 $(y,1^n)$ 时,我们可以调用输入为 $(y,1^m)$ 时的 g 逆变器(ginverter),并输出 g 逆变器返回串的长度在 I 内的最长前缀(例如,如果 g 逆变器返回一个 m 位的串,我们就输出它的一个 n 位长的前缀)。这里 y 是对所有的 $m\in\{n,\ldots,s_I(n)-1\}$,$f(U_n)=g(U_m)$ 值域中的一个值。因此,当输入

$f(U_n)$时对 f 求逆的成功概率等于输入 $g(U_m)$时 g 逆变器的成功概率。问题是用 $m\in\{n,\ldots,s_I(n)-1\}$中的哪一个 m 呢，答案是把每一个都试一下(利用 $s_I(n)=\text{poly}(n)$)。注意：整数被拆分为$[n,\ldots,s_I(n)-1]$形式的区间，每一个区间由惟一的 $n\in I$ 决定。这样，当输入无限多的长度 $m\in\mathbb{N}$ 时，g 逆变器的成功概率就可以转化为当输入无限多的长度 $n\in I$ 时 f 逆变器的成功概率。具体如下，给定一个对 g 求逆的算法 B'，我们可以构造一个对 f 求逆的算法 A'，使得 A'的算法复杂性和成功概率与 B'有关(简单一些，假设对每一个 $y\in\{0,1\}^*$ 和 $m\in\mathbb{N}$ 都有 $B'(y,1^m)\in\{0,1\}^m$；这个假设并不是本质性的，后面关于这一方面的内容我们将用两个脚注来说明)。算法 A'把算法 B'作为一个子程序，当输入 y 和 1^n时(设 y 属于 $f(U_n)$的值域，且 $n\in I$)，A'如下处理：

1. 计算 $s_I(n)$，并设 $k\stackrel{\text{def}}{=}s_I(n)-n-1\geq 0$(因而对每一个 $i=1,\ldots,k$，我们有 $n+i\notin I$)。

2. 对于 $i=0,\ldots,k$，在输入为$(y,1^{n+i})$的情况下，算法 A' 调用算法 B'，从而得到 $z_i\leftarrow B'(y,1^{n+i})$；如果 $g(z_i)=y$，那么 A'输出 n 位长的 z_i的前缀[①]。

注意，对于所有 $x'\in\{0,1\}^n$ 和$|x''|\leq k$，都有 $g(x'x'')=f(x')$，因此如果 $g(x'x'')=y$，则有 $f(x')=y$，这就证实了算法 A'输出的正确性。利用 $s_I(n)=\text{poly}(n)$和 $s_I(n)$在多项式时间算法中是可计算的这一事实，可以进一步得出如下结论：如果 B'是概率多项式时间算法，那么 A'也是。下一步分析 A'的成功概率(表明如果 B'可以在不允许的成功概率下对 g 求逆，那么 A'就可以在不允许的成功概率下对 f 求逆)。

当输入$(g(U_m),1^m)$时，假设 B'以 $\varepsilon(m)$的成功概率对 g 求逆，那么存在 n 使得 $m\in\{n,\ldots,\text{poly}(n)\}$，并且当输入为$(f(U_n),1^m)=(g(U_m),1^m)$时，$A'(f(U_n),1^n)$调用 B'，从而 $A'(f(U_n),1^n)$至少能以 $\varepsilon(m)=\varepsilon(\text{poly}(n))$的概率对 f 求逆。这样 $A'(f(U_n),1^n)$继承了 $B'(g(U_m),1^m)$的成功概率。以下是较为冗长的分析(可以略去不计)[②]。

反过来，假设 g 不是一个强单向函数，设 B'是一个能证明这个矛盾假设的算法。也就是说，存在多项式 $p(\cdot)$，使得对于无限多的 m，B'对 g 求 $g(U_m)$的原像的成功概率至少是$\frac{1}{p(m)}$。记 M 为以上 m 的集合，定义函数 $l_I:\mathbb{N}\rightarrow I$ 使得 $l_I(m)$是 I 中 m 的最大下界(即 $l_I(m)\stackrel{\text{def}}{=}\max\{i\in I:i\leq m\}$)。很明显，对于每一个 m，$m\geq l_I(m)$且 $m\leq s_I(l_I(m))-1$。以下两个结论表明了算法 A'和算法 B'的成功概率之间的联系。

结论 2.2.3.1：设 m 为一整数并且 $n=l_I(m)$，那么

$$\Pr[A'(f(U_n),1^n)\in f^{-1}(f(U_n))]\geq\Pr[B'(g(U_m),1^m)\in g^{-1}(g(U_m))]$$

(即算法 A'对 $f(U_{l_I(m)})$求逆的成功概率以算法 B'对 $g(U_m)$求逆的成功概率为下界。)

证明：在输入$(f(x'),1^n)$时(此处 $x'\in\{0,1\}^n$ 构造算法 A'，对于每一个 $t\in\{n,\ldots,s_I(n)-1\}$算法 A'取值 $B'(f(x'),1^t)$。特别地，由于 $m\geq n$ 且 $m\leq s_I(l_I(m))-1=s_I(n)-1$ 可推出 A'取值 $B'(f(x'),1^m)$。由 g 的定义，对于所有 $x''\in\{0,1\}^{m-n}$，有 $f(x')=g(x'x'')$，从而结论得证。□

结论 2.2.3.2：存在多项式 $q(\cdot)$，使得对于所有的 m，都有 $m<q(l_I(m))$。

证明：设 q 是一个多项式(由 I 的多项式时间算法可数性所规定)，使得 $s_I(n)<q(n)$，则对

---

① 这里，我们假设 $z_i\in\{0,1\}^{n+i}$，指的是，n 是 I 中的最大整数，且最多为 $n+i$。通常，A'输出 z_i 的最长的前缀 x'，并满足$|x'|\in I$。这里 $f(x')=g(z_i)=y$。

② 读者可自己证明：如下分析与 B'的输出长度无关，且不依赖于以前的假设。

于所有 m，都有 $m<s_I(l_I(m))<q(l_I(m))$。□

由结论 2.2.3.2 可知，集合 $S\stackrel{\text{def}}{=}\{l_I(m):m\in M\}$ 是一个无限集（否则的话，集合 S 中所有元素的上限 u 对每一个 $m\in M$，都有 $m<q(l_I(m))\leq q(u)$，这与 M 是无限集的假设相矛盾）。利用结论 2.2.3.1，可进一步得出：对于每一个 $n=l_I(m)\in S$，当输入 $f(U_n)$ 时，A' 对 f 求逆的成功概率至少为：

$$\frac{1}{p(m)}>\frac{1}{p(q(l_I(m)))}=\frac{1}{p(q(n))}=\frac{1}{\text{poly}(n)}$$

此处不等号成立的依据是结论 2.2.3.2。从而可得出 f 不是强单向的，这与最初的假设矛盾。■

2.2.3.2　正则长度(length-regular)和等长度的函数

第二个关于单向函数的有用的约定是，对于每一个 $x,y\in\{0,1\}^*$ 如果 $|x|=|y|$，那么 $|f(x)|=|f(y)|$，则称函数 f 是长度正则的。这里我们指出前面提到的变形(即等式 2.1 和等式 2.2)保持长度的正则性。长度正则性的一个特例是等长度的函数，见等式 2.2。

定义 2.2.4(等长度的函数)：如果对于每一个 $x\in\{0,1\}^*$，都有 $|f(x)|=|x|$，则称函数 f 是等长度的函数。

给定一严格(弱)单向函数 f，可以构造一个等长度的严格(弱)单向函数 f''，如下：设 p 是一个约束长度的、f 扩展的多项式(即 $|f(x)|\leq p(|x|)$)。这样的多项式必定存在，因为 f 是多项式时间可计算的。我们首先构造一个长度正则的函数 f'，定义 f' 如下：

$$f'(x)\stackrel{\text{def}}{=}f(x)10^{p(|x|)-|f(x)|}\tag{2.3}$$

(此处采用 10^* 的形式是为了方便把 $f'(x)$ 分解为 $f(x)$ 和“剩余”的部分。)接下来我们仅对长为 $p(n)+1$ 的串来定义 f''，对于 $n\in\mathbb{N}$，令

$$f''(x'x'')\stackrel{\text{def}}{=}f'(x')\text{，这里 }|x'x''|=p(|x'|)+1\tag{2.4}$$

显然，f'' 是等长度的函数。

命题 2.2.5：如果 f 是强(弱)单向函数，那么 f' 和 f''(分别见等式 2.3 和等式 2.4 的定义)也是强(弱)单向函数。

略证：容易看出 f' 和 f'' 都是多项式时间可计算的。利用与前述证明中类似的可约性理论(reducibility argument)，可以证明 f' 和 f'' 的难求逆性。例如，给定对 f' 求逆的算法 B'，我们可以构造对 f 求逆的算法 A' 如下：当输入 y 和 1^n(假定 y 是 $f(U_n)$ 值域中的一个)时，算法 A' 以输出 $B'(y10^{p(n)-|y|},1^n)$ 终止。■

减少辅助输入 $1^{|x|}$　读者能够很容易地证实：如果 f 是等长度的，那么再为求逆算法提供辅助输入 $1^{|x|}$($f(x)$ 除外)就是多余的。如果 f 是长度正则的，并且不以大于一个多项式的阶收敛它的输入(即存在多项式 $p(\cdot)$，使得对于所有的 x，都有 $p(|f(x)|\geq|x|)$ 时，会有同样的结论。在今后，我们只处理长度正则的并且不以大于一个多项式的阶收敛它的输入的单向函数。此外，我们将大量讨论等长度的单向函数。在所有这些情况下，为了不失一般性，我们都假设对每一个求逆算法只有 $f(x)$ 作为输入。

1-1 的单向函数　如果 f 是 1-1 的，则 f′(等式 2.3 定义)也是 1-1 的，但 f″(等式 2.4 定义)不是。因此，当给定一个 1-1 的函数时，为了不失一般性，我们只能假设它是长度正则的，而不能假设它是等长度的。此外，假设存在 1-1 的单向函数似乎比假设存在任意的(当然可以是等

长度的)单向函数更健壮。进一步讨论详见 2.4 节。

2.2.4 单向函数的候选形式

下面是几种单向函数的候选形式。显然,我们不知道这些函数是否确实是单向的。这仅仅是我们所做的大量研究的一种猜想,而针对这些形式,到目前为止还没有找到一个有效的求逆算法(有能引起注意的成功概率的算法)。

2.2.4.1 整数分解

除了对构造可行的整数分解算法所做的大量研究,我们所知道的分解整数的最好算法是具有亚指数的运行时间的算法。因此,认为函数 f_{mult} 是单向的是合理的,如果函数 f_{mult} 将它的输入串分为两部分并返回与这些部分(代表整数)相乘的整数结果(二进制表示)。即,令

$$f_{\mathrm{mult}}(x,y)=x\cdot y$$

这里 $|x|=|y|$,并且 $x\cdot y$ 表示整数 x 和 y(由串来表示)相乘所得的整数(由串来表示)。显然,f_{mult} 是多项式时间可计算的。假设分解是很困难的(例如,给定两个均匀选择的 n 位长的素数的乘积,再找到它的素因子是不可行的),并利用素数稠密性理论(指出比 N 小的整素数至少有 $\frac{N}{\log_2 N}$ 个),可推导出 f_{mult} 至少是弱单向的(进一步的讨论见习题 8)。其他一些有关整数分解的常见函数(例如 RSA 函数)将在 2.4.3 小节中讨论。

2.2.4.2 译随机线性码

在纠错码领域,最著名的一个公开问题是寻找随机线性码解译算法。特别需要注意的是具有固定的信息率的随机线性码,可纠正固定大小的错误。一个(n,k,d)线性码是一个二进制矩阵,矩阵的任意行非空子集的模 2 矢量和得到新的矢量,它至少有 d 个 1(对一个 k 位长的消息编码,就是把它乘以这个 $k\times n$ 矩阵,得到一个 n 位长的向量,该向量即使有$\frac{d}{2}$个位发生错误,仍可以得到惟一正确的原象)。只要$\frac{k}{n}<1-H_2(\frac{d}{n})$,线性码的 Gilbert-Varshamov 界保证了这样一种码的存在,这里当 $p<\frac{1}{2}$时,$H_2(p)\stackrel{\mathrm{def}}{=}-p\log_2 p-(1-p)\log_2(1-p)$

反之,$H_2(p)\stackrel{\mathrm{def}}{=}1$

(即,$H_2(\cdot)$)是二进制熵函数的一个变形)。类似地,如果存在 $\epsilon>0$,使得$\frac{k}{n}<1-H_2(\frac{(1+\epsilon)d}{n})$成立,那么几乎所有的 $k\times n$ 二进制矩阵都能构成一个(n,k,d)线性码。设有 3 个常数 $k,\delta,\epsilon>0$,满足 $k<1-H_2((1+\epsilon)\delta)$,则函数 f_{code} 作为单向函数的一个候选形式似乎是合理的:

$$f_{\mathrm{code}}(C,x,i)\stackrel{\mathrm{def}}{=}(C,xC+e(i))$$

这里 C 是一个 $kn\times n$ 的二进制矩阵,x 是一个 kn 维二进制向量,而 i 是一个 n 维二进制向量的序数,这个向量在这一类相应的可列集中至多有$\frac{\delta n}{2}$个 1(向量本身记为 $e(i)$),所有计算都是在 n 维二进制向量空间中进行。显然,只要选择有效向量可列集,f_{code} 就是多项式时间可计算的。对 f_{code} 求逆的有效算法可以生成一个对有固定信息率的线性码的不可忽略部分解码的有效算法(这将是编码理论的震撼全球的结论)。

2.2.4.3 子集和问题

考虑函数 f_{ssum}，其定义如下：

$$f_{\text{ssum}}(x_1,\ldots,x_n,I)=(x_1,\ldots,x_n,\sum_{i\in I}x_i)$$

这里 $|x_1|=\ldots=|x_n|=n, I\subseteq\{1,2,\ldots,n\}$。显然，$f_{\text{ssum}}$ 是多项式时间可计算的。子集和问题是 **NP**-完全的这一事实不能作为 f_{ssum} 单向性的证据。另一方面，子集和问题在特定场合又是非常简单的(例如具有"隐藏结构"或/和"低密度"的情况)，所以不能排除这种可能性。我们猜想 f_{ssum} 是单向的，是建立在不知道处理随机"高密度"事件(即，像 f_{ssum} 定义中所表述的事件中元素的长度约等于它们的数量)的算法的基础上。

2.2.5 非均匀单向函数

在前面两个单向函数的定义中，求逆算法都是概率多项式时间算法。两种定义的较强的形式都要求函数不能被非均匀多项式长度电路族求逆。我们强调易于计算的条件仍以均匀算法的形式给出。例如，下面是一个强(等长度)单向函数的非均匀形式的定义。

定义 2.2.6(非均匀强单向函数)：如果满足以下两个条件：

1. 容易计算：存在一个多项式时间算法 A，使输入为 x 时输出为 $f(x)$。

2. 求逆困难：对于每一个(即使是非均匀的)多项式长度电路族 $\{C_n\}_{n\in N}$、每一个正多项式 $p(\cdot)$ 和所有足够大的 n，都存在

$$\Pr[C_n(f(U_n))\in f^{-1}(f(U_n))]<\frac{1}{p(n)}$$

则称函数 $f:\{0,1\}^*\to\{0,1\}^*$ 为非均匀单向函数。

第二个条件中的概率只适用于所有 U_n 的可能值。此处我们要指出所有非均匀单向函数均为单向函数(也就是说在非均匀意义上的单向函数)。

命题 2.2.7：如果 f 是非均匀单向函数，那么它一定是单向函数。也就是说，如果 f 满足定义 2.2.6，那么它也满足定义 2.2.1。

证明：我们可以在不降低成功概率的情况下，将任何一个(均匀)概率多项式时间算法的求逆算法转换成一个非均匀多项式长度电路族。这服从转换定律(meta-theorem)(见 1.3.3 节)。具体如下：

假设 A' 是一个概率多项式时间(求逆)算法。r_n 代表能使 A' 的成功概率(每一个输入为 $f(U_n)$ 概率的平均值)最大化的一个投币序列。也就是说，r_n 满足下式：

$$\Pr[A'_{r_n}(f(U_n))\in f^{-1}(f(U_n))]\geq\Pr[A'(f(U_n))\in f^{-1}(f(U_n))]$$

式中第一个概率只适用于所有 U_n 的可能值，第二个概率也适用于所有 A' 的投币可能序列。($A'_r(y)$ 表示算法 A' 在输入为 y 和中间投币值 r 时的输出)。设计的电路 C_n 包含算法 A' 和 r_n 序列的代码(r_n 的长度是 n 的多项式)。■

此处我们指出：通常情况下，平均值方法(采用前面应用的形式)使我们能够将概率多项式时间算法转换成非均匀多项式长度电路族。因而总体上讲，安全的非均匀概念(即针对非均匀多项式长度电路族的健壮性)包含了安全的均匀概念(即针对概率多项式时间算法的健壮性)。这种转换不一定非成立不可。特别地，有可能存在这样的情况：单向函数存在(在均匀意义

下），而不存在非均匀单向函数。但是，这种情况（即单向函数只在均匀意义下存在）出现的可能性很小，并且大家普遍认为非均匀单向函数是存在的。事实上，所有前述的算法均被认为是非均匀单向函数。

2.3 弱单向函数隐含强单向函数

首先我们必须指出，并非每个弱单向函数都是强单向函数。例如，假设有一个单向函数 f（为了不失一般性，假定 f 是等长度的），利用 f 来构造函数 g：设 $|p|=|x|$，那么如果 p 以 $\log_2|x|$ 个零为开始，$g(p,x)=(p,f(x))$；否则，$g(p,x)=(p,x)$[①]。显然，g 不是强单向函数（因为对长为 $2n$ 的序列来说，除了 $\frac{1}{n}$ 部分，函数 g 等价于恒等函数）。为了证明 g 是一个弱单向函数，我们使用一个可约性理论。

命题 2.3.1：设 f 是一个单向函数（即使是在弱的意义下），则上文构造的函数 g 是一个弱单向函数。

证明：直观上说，根据非恒等变换输入，对 g 求逆是和对 f 求逆有关的。如果对于长为 $2n$ 的输入，函数 g 可以求逆的概率大于 $1-\frac{1}{n}$，那么根据 g 作用于 f 的输入，g 可以求逆的概率是很明显的。因此，如果 g 不是弱单向函数，那么 f 也不是。完整的证明过程如下：

假定有一个关于对 g 求逆的概率多项式时间算法 B'，再构造一个概率多项式时间算法 A'，表示能以"相关"成功概率对 f 求逆。当输入 y 时，算法 A' 规定 $n\stackrel{\text{def}}{=}|y|$，$l\stackrel{\text{def}}{=}\log_2 n$，$p'$ 一律在 $\{0,1\}^{n-l}$ 中均匀选择，计算 $z\stackrel{\text{def}}{=}B'(0^l p',y)$ 直至 z 输出 n 位，设 S_{2n} 为所有以 $\log_2 n$ 个零作为开始的长为 $2n$ 比特的序列（即 $S_{2n}\stackrel{\text{def}}{=}\{0^{\log_2 n}\alpha:\alpha\in\{0,1\}^{2n-\log_2 n}\}$）。那么，通过构造 A' 和 g，可以得到

$$\begin{aligned}
&\Pr[A'(f(U_n))\in f^{-1}(f(U_n))]\\
&\geq \Pr[B'(0^l U_{n-l},f(U_n))\in(0^l U_{n-l},f^{-1}(f(U_n)))]\\
&= \Pr[B'(g(U_{2n}))\in g^{-1}(g(U_{2n}))\mid U_{2n}\in S_{2n}]\\
&\geq \frac{\Pr[B'(g(U_{2n}))\in g^{-1}(g(U_{2n}))]-\Pr[U_{2n}\notin S_{2n}]}{\Pr[U_{2n}\in S_{2n}]}\\
&= n\cdot\left(\Pr[B'(g(U_{2n}))\in g^{-1}(g(U_{2n}))]-\left(1-\frac{1}{n}\right)\right)\\
&= 1-n\cdot(1-\Pr[B'(g(U_{2n}))\in g^{-1}(g(U_{2n}))])
\end{aligned}$$

（在第二个不等式中，我们利用了 $\Pr[A|B]=\frac{\Pr[A\cap B]}{\Pr[B]}$ 和 $\Pr[A\cap B]\geq\Pr[A]-\Pr[\neg B]$ 这两个关系式。）上述表达式只在 $\Pr[B'(g(U_{2n}))\in g^{-1}(g(U_{2n}))]>1-\frac{1}{n}$ 的条件下才有意义。

对每个多项式 $p(\cdot)$ 和每个整数 n，如果在输入 $g(U_{2n})$ 时用 B' 对 g 求逆成功的概率大于 $1-\frac{1}{p(2n)}$，那么当输入 $f(U_n)$ 时用算法 A' 对 f 求逆成功的概率大于 $1-\frac{n}{p(2n)}$。因此，如果 g 不是弱单向函数（即：对每个多项式 $p(\cdot)$，一定存在许多这样的 m，使得输入 $g(U_m)$ 时能够对 g 求逆的概率 $\geq 1-1/p(m)$），那么 f 也不是一个弱单向函数（即：对每个多项式 $q(\cdot)$，一定存在许多个这样的 n，使得输入 $f(U_n)$ 时能够对 f 求逆的概率 $\geq 1-1/q(n)$，其中 $q(n)=p(2n)/$

---

① 在本文中，我们把 $\log_2|x|$ 看作一个整数。严密的证明必须用 $\lceil\log_2|x|\rceil$ 代替 $\log_2|x|$，并做一些微小的调整。

n)。这与假设(f 是一个弱单向函数)矛盾。

概括说来,假如有一个概率多项式时间算法,利用该算法在输入 $g(U_{?n})$时对 g 求逆的成功概率是 $1-\frac{1}{n}+\alpha(n)$,那么我们必然能得到这样一个概率多项式算法,当输入 $f(U_n)$时用该算法对 f 求逆的成功概率是 $n\cdot\alpha(n)$。这样,只要 f 是一个(弱)单向函数,对某些多项式 q 来说 $n\cdot\alpha(n)<1-1(1/q(n))$一定成立,因此 g 也一定是一个弱单向函数(因为当输入 $g(U_{2n})$时,每个试图对 g 求逆的概率多项式时间算法失败的概率至少是 $\frac{1}{n}-\alpha(n)>\frac{1}{n\cdot q(n)}$)。■

前面我们已经证明了:除非单向函数不存在,否则一定存在不是强单向函数的弱单向函数。这就排除了所有单向函数都是强单向函数的可能性,同时也排除了所有单向函数都(只)是弱单向函数的可能性。特别地,弱单向函数的存在性暗示了强单向函数的存在性。

定理 2.3.2:弱单向函数存在当且仅当强单向函数存在。

我们强烈建议读者不要跳过 2.3.1 小节中的证明过程,因为此证明过程对理解后文有指导意义,而且此证明过程说明了计算难题的放大(amplification)比类似的概率事件的放大难得多。2.3.3 小节将进一步讨论这两个方面的问题。2.3.2 小节给出了一个"有趣的"例子来说明此证明过程。(事实上,许多读者倾向于先读 2.3.2 小节再读 2.3.1 小节。)

2.3.1 定理 2.3.2 的证明

设 f 是一个弱单向函数,p 是一个由弱单向函数的定义保证的多项式。也就是说,每个概率多项式时间算法当输入 $f(U_n)$时不能对 f 求逆的概率至少等于$\frac{1}{p(n)}$。为了把问题简化,我们假设 f 是等长度的(即,对所有 x 都有 $|f(x)|=|x|$)。这个假设不是必需的,但是命题 2.2.5 可以证明它是合理的。定义一个函数 g 如下:

$$g(x_1,\ldots,x_{t(n)})\stackrel{\text{def}}{=}f(x_1),\ldots,f(x_{t(n)}) \tag{2.5}$$

其中,$|x_1|=\cdots=|x_{t(n)}|=n$,且 $t(n)\stackrel{\text{def}}{=}n\cdot p(n)$。即 g 的输入长为 $n^2p(n)$比特,把它分成 $t(n)$段,每段长为 n,然后 f 作用于每段。

显然,g 可以在多项式时间内计算出来(通过一个算法把输入序列分段,再使 f 作用于每段)。而且不难看出,当输入 $g(x_1,\ldots,x_{t(n)})$时,如果要对 g 求逆就必须找到每个 $f(x_i)$的原像。不难推导出 g 是一个强单向函数。通过隐含的假设,我们可以继续得到,求逆算法是对于每个 $f(x_i)$分别进行的。如果这个假设成立,那么某一求逆算法能够对所有 $f(x_i)$求逆的概率最多是$(1-\frac{1}{p(n)})^{n\cdot p(n)}<2^{-n}$(这一概率同样是可以忽略的)。然而,我们还没有证明对 g 求逆的算法独立于每个 $f(x_i)$这一假设是正确的。因此,还需要更复杂的证明。

下面是我们的证明方法的要点。我们用反证法来证明 g 是强单向函数。假设 g 不是强单向函数,即假设存在这样一个多项式时间算法,用此算法对于 g 求逆的概率是不可忽略的。这样可以得到一个多项式时间算法,此算法对于无限多个 n 来说,当输入 $f(U_n)$ 时能对 f 求逆的概率大于 $1-\frac{1}{p(n)}$,这与我们的假设矛盾。对 f 的求逆算法是将对 g 的求逆算法作为子程序(无需假设后一算法的运行方式)。(需要强调的是,我们假设 g 逆变器不是以特殊的方式,而是以一般的方式来构造 f 逆变器。)详细证明如下。

假设 g 不是强单向函数。由定义可知,存在一个概率多项式时间算法 B'和一个多项式 $q(\cdot)$,对无限多个 m,都有

$$\Pr[B'(g(U_m)) \in g^{-1}(g(U_m))] > \frac{1}{q(m)} \tag{2.6}$$

设 M' 是所有满足这一关系的无穷整数序列，N' 是所有满足 $n^2 \cdot p(n) \in M'$ 这一条件的 n 所构成的无穷序列（请注意对某一整数 n 来说，我们所说的 m 都是 $n^2 \cdot p(n)$ 形式的）。

通过 B'，可以构造一个概率多项式时间算法 A' 来对 f 求逆。给定一个输入 y（假设在 f 的值域内），输入 $a(|y|)$ 次 y，算法 A' 按以下概率进程（记为 I）进行，其中 $a(\cdot)$ 是一个由多项式 p 和 q 决定的多项式（特别地，令 $a(n) \stackrel{\text{def}}{=} 2n^2 \cdot p(n) \cdot q(n^2 p(n))$）。

进程 I

输入：y（记 $n \stackrel{\text{def}}{=} |y|$）。

从 $i=1$ 到 $t(n)$ 循环以下步骤：

1. 独立均匀地选取一系列字符串 $x_1, \ldots, x_{t(n)} \in \{0,1\}^n$。

2. 计算 $(z_1, \ldots, z_{t(n)}) \leftarrow B'(f(x_1), \ldots, f(x_{i-1}), y, f(x_{i+1}), \ldots, f(x_{t(n)}))$（请注意在第 i 位上用 y 代替 $f(x_i)$）。

3. 若 $f(z_i) = y$，则停止并输出 z_i（这就被视为是成功的）。

利用 2.6 式，我们现在可以提出一个算法 A' 成功的概率的下界。为此，定义一个集合，记为 S_n，它包含所有 n 比特的字符串，对这些字符串进程 I 成功的概率（特别地，大于 $\frac{n}{a(n)}$）是不可忽略的。（这个概率值只能在进程 I 中以投硬币的方式取值）。即为，

$$S_n \stackrel{\text{def}}{=} \left\{ x : \Pr[I(f(x)) \in f^{-1}(f(x))] > \frac{n}{a(n)} \right\}$$

在下面的两个论题中，我们将证明：S_n 至多包含 $\frac{1}{2p(n)}$ 的长为 $n \in N'$ 的字符串 $x \in S_n$，对每个字符串，当输入 $f(x)$ 时用算法 A' 对 f 求逆的成功概率以指数速度接近于 1。从而，对 $n \in N'$，当输入 $f(U_n)$ 时，算法 A' 对 f 求逆的成功概率大于 $1-\frac{1}{p(n)}$，这与我们的假设矛盾。

结论 2.3.2.1：对每个 $x \in S_n$，有

$$\Pr[A'(f(x)) \in f^{-1}(f(x))] > 1 - \frac{1}{2^n}$$

证明：根据集合 S_n 的定义可得，进程 I 可以对 $f(x)$ 求逆的概率至少是 $\frac{n}{a(n)}$。算法 A' 只是重复过程 I $a(n)$ 次，所以

$$\Pr[A'(f(x)) \notin f^{-1}(f(x))] < \left(1 - \frac{n}{a(n)}\right)^{a(n)} < \frac{1}{2^n}$$

论题成立。

结论 2.3.2.2：对每个 $n \in N'$，有

$$|S_n| > \left(1 - \frac{1}{2p(n)}\right) \cdot 2^n$$

证明：假设其逆命题 $|S_n| \le (1-\frac{1}{2p(n)}) \cdot 2^n$ 成立。我们可以得到与 2.6 式（即，我们关于 B' 成功概率的假设）矛盾的结论。据此前提（对 $n \in N0$）有，

$$s(n) \stackrel{\text{def}}{=} \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)}))] > \frac{1}{q(n^2 p(n))} \tag{2.7}$$

设 $U_n^{(1)}, \ldots, U_n^{(n^2 \cdot p(n))}$ 为随机变量 $U_{n^2} p(n)$ 中长为 n 比特的段（即每个 $U_n^{(i)}$ 是 $\{0,1\}^n$ 中均匀分布的独立的随机变量）。根据 $U_n^{(i)}$ 是否属于 S_n，我们可以把 2.7 式中考虑的事件分为两类不相关的事件。直观上说，B' 在这种情况下的性能不是很好，因为这种情况与输入 S_n 的原像时

I 的成功概率相符。另一方面，所有 $U_n^{(i)}$ 都属于 S_n 的概率是很小的。特别地，我们定义

$$s_1(n) \stackrel{\text{def}}{=} \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \wedge (\exists i \text{ s.t. } U_n^{(i)} \notin S_n)]$$

$$\text{和 } s_2(n) \stackrel{\text{def}}{=} \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \wedge (\forall i : U_n^{(i)} \in S_n)]$$

显然，$s(n)=s_1(n)+s_2(n)$（因为各 s_i 事件是不相关的）。接下来我们给出 $s_1(n)$ 和 $s_2(n)$ 的上界，二者之和小于 2.7 式中给出的 $s(n)$ 的下界，于是产生矛盾。

首先，我们计算 $s_1(n)$ 的上界。当输入 $f(x)$ 时算法 I 对 f 求逆的成功概率是和 B' 的成功概率有关的，B' 是在输入随机的 f 的映像（包含 $f(x)$）时对 g 求逆的算法。特别地，对每个 $x \in \{0,1\}^n$ 和每个 $1 \leq i \leq n \cdot p(n)$，当输入 $f(x)$ 时 I 对 f 求逆的成功概率大于或等于在 $U_n^{(i)}=x$ 的条件下 B' 在输入 $g(U_{n^2 p(n)})$ 时对 g 求逆的成功概率（因为 B' 能对 g 求逆意味着输入第 i 段时可以对 f 求逆，这包括在 I 的成功概率内）。于是对每个 $x \in \{0,1\}^n$ 和每个 $1 \leq i \leq n \cdot p(n)$，有

$$\begin{aligned}&\Pr[I(f(x)) \in f^{-1}(f(x))]\\&\quad \geq \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \mid U_n^{(i)} = x]\end{aligned} \tag{2.8}$$

由于对 $x \notin S_n$ 来说上式左边不能太大，因此我们要证明（右边）$s_1(n)$ 也不可能太大。特别地，利用 2.8 式可得，

$$\begin{aligned}
s_1(n) &= \Pr[\exists i \text{ s.t. } B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \wedge U_n^{(i)} \notin Sn]\\
&\leq \sum_{i=1}^{n \cdot p(n)} \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \wedge U_n^{(i)} \notin S_n]\\
&\leq \sum_{i=1}^{n \cdot p(n)} \sum_{x \notin S_n} \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \wedge U_n^{(i)} = x]\\
&= \sum_{i=1}^{n \cdot p(n)} \sum_{x \notin S_n} \Pr[(U_n^{(i)} = x] \cdot \Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \mid U_n^{(i)} = x]\\
&\leq \sum_{i=1}^{n \cdot p(n)} \max_{x \notin S_n}\{\Pr[B'(g(U_{n^2 p(n)})) \in g^{-1}(g(U_{n^2 p(n)})) \mid U_n^{(i)} = x]\}\\
&\leq \sum_{i=1}^{n \cdot p(n)} \max_{x \notin S_n}\{\Pr[I(f(x)) \in f^{-1}(f(x))]\}\\
&\leq n \cdot p(n) \cdot \frac{n}{a(n)} = \frac{n^2 \cdot p(n)}{a(n)}
\end{aligned}$$

（最后一个不等式利用了 S_n 的定义，前一个式子用到了 2.8 式。）

下面计算 $S_2(n)$ 的上界。根据否命题 $|S_n| \leq (1-\frac{1}{2p(n)}) \cdot 2^n$，可得

$$\begin{aligned}
S_2(n) &\leq \Pr[\forall i : U_n^{(i)} \in S_n]\\
&\leq (1-\frac{1}{2p(n)})^{n \cdot p(n)}\\
&< \frac{1}{2^{n/2}} < \frac{n^2 \cdot p(n)}{a(n)}
\end{aligned}$$

（最后一个不等式对足够大的 n 成立。）

结合 S_i 的上界，可得 $S_1(n)+S_2(n)<\frac{2n^2 \cdot p(n)}{a(n)}=\frac{1}{q(n^2 p(n))}$（根据 $a(n)$ 的定义可知等式成立）。然而，根据 2.7 式又可得 $s_1(n)+s_2(n)=s(n)>\frac{1}{q(n^2 p(n))}$。于是产生矛盾，论题得证。□

综合结论 2.3.2.1 和 2.3.2.2，可得

$$\begin{aligned}
&\Pr[A'(f(U_n)) \in f^{-1}(f(U_n))] \\
&\quad \geq \Pr[A'(f(U_n)) \in f^{-1}(f(U_n)) \wedge U_n \in S_n] \\
&\quad = \Pr[U_n \in S_n] \cdot \Pr[A'(f(U_n)) \in f^{-1}(f(U_n)) \mid U_n \in S_n] \\
&\quad \geq (1 - \frac{1}{2p(n)}) \cdot (1 - 2^{-n}) > 1 - \frac{1}{p(n)}
\end{aligned}$$

这就证明了存在一个概率多项式时间算法(即 A'),此算法对 $n \in N'$ 有输入 $f(U_n)$ 对 f 求逆的成功概率大于 $1-\frac{1}{p(n)}$。根据我们的假设,这个结论可以证明 g 不是一个强单向函数(即等式 2.6),但这与每个概率多项式时间算法不能对 f 求逆的概率至少为 $\frac{1}{p(n)}$ 的假设相矛盾。于是定理成立。

2.3.2 一个有趣的例子

下面我们进一步说明定理 2.3.2 的证明过程中的算法的思想。首先,定量分析弱单向函数的概念。如果对所有概率多项式时间算法 A' 和所有有限多个 n,当输入 $f(U_n)$ 时算法 A' 不能找到其在 f 作用下的原像的概率至少是 $\rho(n)$,那么我们可以称这个(可计算的多项式时间) f 是 ρ 单向的。(对某个多项式 p 来说,每个弱单向函数是 $1/p()$ 单向的,而强单向函数是 $(1-\mu())$ 单向的,其中 μ 是可忽略的函数。)

命题 2.3.3(有趣的例子):设 f 是 $\frac{1}{3}$ 单向的,令 $g(x_1,x_2) \stackrel{\text{def}}{=} (f(x_1)f(x_2))$,则 g 是 0.55 单向的(其中 $0.55 < 1-(\frac{2}{3})^2$)。

证明要点:用反证法证明。假设存在一个多项式时间算法 A',该算法对有限多个 n 来说对 $g(U_{2n})$ 求逆成功的概率大于 $1-0.55=0.45$。对任意 n,令 $N \stackrel{\text{def}}{=} 2^n$,为了将问题简化,假设 A' 是确定性的。假设有一 $N \times N$ 阶矩阵,其每项对应于序对 $(x_1,x_2) \in \{0,1\}^n \times \{0,1\}^n$,如果在输入 $g(x_1,x_2)=(f(x_1),f(x_2))$ 时 A' 能成功地对 g 求逆,就把 (x_1,x_2) 项置为 1;否则置为 0。我们提出的与定理矛盾的假设就是在这个矩阵中置为 1 的项所占的百分数大于 45%。

原始(未经证明的)假设是算法 A' 独立作用于每个序对 $(f(x_1),f(x_2))$ 上。如果真是那样,那么 A' 成功的区域就是一个没有显著特点的矩形 $R \times C \subseteq \{0,1\}^n \times \{0,1\}^n$(即对应于所有序对 (x_1,x_2),其中 $x_1 \in R, x_2 \in C$,对应于 $R \subseteq \{0,1\}^n, C \subseteq \{0,1\}^n$)。根据 f 是 $\frac{1}{3}$ 单向这一前题可得 $|R|,|C| \leq \frac{2}{3}$,因此 $\frac{|R \times C|}{N^2} \leq \frac{4}{9} < 0.45$,这就与我们关于算法 A' 的假设相矛盾。

然而如前所述,原始假设是未经证实的,因此尚需更复杂的证明。一般说来,A' 的成功区域记为 S,可能是满足 $|S| > 0.45 \cdot N^2$ 的 $\{0,1\}^n \times \{0,1\}^n$ 的任意子集(由假设可知)。如果某一行 x_1 至少包含 0.1% 的 1 项,那么这行就是好的情况(某一列 x_2 可类似定义);反之则是坏的情况(见图 2.2)。该证明过程的主要算法部分确定了结论 2.3.3.1

结论 2.3.3.1:好的行(或列)至多占 66.8%。

一旦此结论得证,剩下的就是简单的组合数学(即计算)问题了。即计算 S 的面积的上界,分别通过计算好的行和好的列的交点中 1 的数目以及坏的行和坏的列的交点中 1 的数目得到的:根据结论 2.3.3.1 可得好的行和好的列的交点中最多有 $(0.668N)^2$ 项,由定义可得每个坏的行(坏的列类似)中 1 的数目最多为 $0.001N$。于是,$|S| \leq (0.668N)^2 + 2 \cdot N \cdot 0.001N < 0.449 \cdot N^2$,这与我们的假设(即 $|S| > 0.45 \cdot N^2$)矛盾。

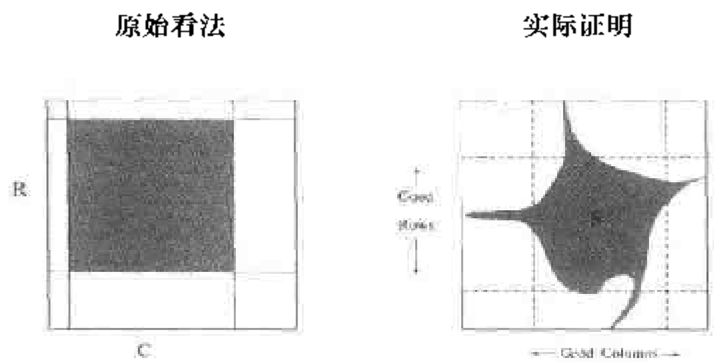


图 2.2 命题 2.3.3 的原始看法与实际证明

结论 2.3.3.1 的证明：假设命题不成立，则好的行所占的百分数大于 66.8%（列的证明类似）。为了得到矛盾，我们可以构造一个算法按如下步骤对 f 求逆。对于给定输入 y，算法重复以下步骤 10 000 次：

1. 在 $\{0,1\}^n$ 中均匀地选取 x_2。
2. 当输入 $(y, f(x_2))$ 时启动算法 A'，得到输出 (x', x'')。
3. 若 $f(x') = y$ 则停止，输出 x'。

显然，此算法在多项式时间内有效，剩下的工作就是分析此算法对 f 求逆的成功概率。

对每个好的 x_1，此算法对给定输入 $y = f(x_1)$ 能够对 f 求逆的概率至多是 $(1-0.001)^{10\,000} < 0.001$。于是，对于给定输入 $f(U_n)$，此算法对 f 求逆成功的概率至少是 $0.668 \cdot 0.999 > \frac{2}{3}$，这与假设 f 是 $\frac{1}{3}$ 单向函数相矛盾。

2.3.3 讨论

2.3.3.1 可约性理论：摘要

下面让我们回顾一下定理 2.3.2 的证明过程的结构。给定一个弱单向函数 f，我们首先构造一个可在多项式时间内计算的函数 g，以便在后面证明 g 是一个强单向函数。为了证明 g 是一个强单向函数，我们使用了可约性理论。此证明方法把假设的与 g 是强单向函数相矛盾的有效算法转化为与前提 f 是弱单向函数相矛盾的有效算法，从而 g 一定是一个强单向函数。需要强调的是，我们的算法转换其实是一个随机化的库克（Cook）变形[①]，不需要对预期算法（用来对 g 求逆）的结构做任何隐含或直接的假设。我们无法证明某些假设（例如，对每个给定的段对 g 求逆是独立的这一"自然"假设）的正确性（至少在目前，基于我们对有效计算的性质的理解下，是不能的）。

我们用术语可约性理论，而不单单说变形，是为了强调我们这里谈及的不是标准（最坏情况复杂度）的变形。下面让我们说明一下二者的区别：我们为了简化解决一个问题的任务而考虑解决另一问题的任务；也就是说，为了构造一个解决第一个任务的进程我们使用一个解决第

---

① 将一个计算问题 Π_1 变换为另一个问题 Π_2 的（随机）Cook 变形，是求解 Π_1 并作为 Π_2 的质询的（概率）多项式时间预言机。

二个任务的进程。然而，标准变形是假设第二个任务包括一个适用于所有条件(即在最坏情况下)的理想进程，并且可以为第一个任务构造这样一个进程。这样，在每个“非典型的”情况下，变形可能都要调用(用来解决第二个任务的)给定的进程。这在我们的可约性理论中是不可能出现的。这里，我们给出了一个进程，该进程可以以满足一定概率分布的概率来解决第二个任务。于是，应用可约性理论时，我们不能在任何场合都调用此进程。相反，必须考虑经变形简化的第二个任务的概率分布。大多数情况下，后者的概率分布等同于前提(关于解决第二个任务的可能性)所谈及的概率分布，其他情况也可以处理(例如，这些概率分布可能由于某一特殊原因“十分接近”)。无论如何，都应当仔细分析可约性理论引起的概率分布。

2.3.3.2 与信息论的类比

定理 2.3.2 有个自然的信息论(或“概率”)的类比，它断言：重复一个具有不可忽略的失败概率的实验足够多次，实验失败的概率就会很高。读者可能已经注意到定理 2.3.2 的证明比此信息论的类比的证明复杂得多。在信息论上下文中，重复事件被定义为相互独立的事件，而在我们可计算性的上下文中，这种独立性(对应于在定理 2.3.2 证明之初给出的原始证明)是不可保证的。这两种系统的另一不同之处在于：信息论中，没有任何失败事件发生的概率会随着重复次数按指数递减；相反，在可计算性中，我们只能得到有关多项式时间算法的求逆概率的不确定但可忽略的界。而且，定理 2.3.2 证明过程中构造的 g 对给定的输入 $g(U_{n^2 p(n)})$ 能够成功求逆的概率是按亚指数递减的(譬如，按概率 $2^{(-\log_2 n)^3}$ 递减)，而在信息论中按指数递减(即 e^{-n})。

2.3.3.3 弱单向函数与强单向函数：总结

根据定理 2.3.2 可知，当我们假设单向函数存在时，不必特别指出是强单向函数还是弱单向函数。也就是说，单就单向函数的存在性而言，弱单向函数和强单向函数的概念是等同的。但是，从有效性的角度考虑，二者并非等同，因为以上由弱单向函数到强单向函数的变换是不切实际的。其他更有效的变换对单向置换和其他特殊类型的单向函数来说是存在的。感兴趣的读者可以参考 2.6 节。

2.4 单向函数的多样性

本节我们主要讨论有关单向函数的几个问题。第一小节我们假设单向函数存在，从而给出了一个(强)单向函数。此函数的构造是个严密而抽象的问题。相反，其他小节讨论的问题则具有实际意义。首先，我们介绍了单向函数的另一种描述方式(formulation)，这种方式更适合于描述有许多固有的候选形式(candidate)的单向函数，而且在描述一些单向函数的常用形式时我们确实使用这种方式。接着，我们用这种方式给出一些有附加属性的单向函数；特别地，我们考虑(单向)陷门置换(trapdoor permutation)和 无爪 claw-free 函数对。需要注意的是，在本书其他章节提出的几个机制的构造中都使用了这些附加属性(例如，陷门置换用于公钥加密体制的构造，而无爪 置换用于无冲突散列函数的构造)。最后我们用有关提出单向函数候选形式的“艺术”的评论来总结本节。

2.4.1* 通用单向函数

利用通用机的概念和上一节的结论，证明通用单向函数的存在性是可能的；也就是说倘若单向函数存在，我们就能找出一个函数是单向函数。

命题 2.4.1：多项式时间可计算的(强)单向函数存在当且仅当单向函数存在。

证明要点：关键结论是单向函数存在当且仅当可用二次项时间算法评估的单向函数存在(特殊时间边界的选择并不重要；重要的是这样一个特殊的时间边界存在)。我们利用构造性方法(padding argument)来证明这个结论。详细证明如下。

令 f 是任意单向函数，$p(\cdot)$ 是一个限制计算 f 的算法的时间复杂度的多项式。定义 $g(x'x'')\stackrel{\text{def}}{=}f(x')x''$，其中 $|x'x''|=p(|x'|)$。计算 g 的算法首先把输入分解成 x' 和 x''，于是 $|x'x''| = p(|x'|)$，然后 f 作用于 x'。分解和其他额外操作可以在二次项时间($|x'x''|$)的函数内完成，而计算 $f(x')$ 可以在 $p(|x'|) = |x'x''|$(其输入长度是线性的)这一时间内完成。因此，(利用图灵机)g 可以在二次项时间内被计算出来。读者可以利用"可约性理论"(与命题 2.2.5 的证明类似)来验证 g 是单向函数。

下面我们给出一个(通用单向)函数，记为 f_{uni}：

$$f_{\text{uni}}(\text{desc}(M),x)\stackrel{\text{def}}{=}(\text{desc}(M),M(x)) \tag{2.9}$$

其中 desc(M)表示一种图灵机 M，如果当输入 x 时，M 在二次项时间内运行，则 $M(x)$表示输入 x 时 M 的输出，否则 $M(x)$就等于 x(为了不失一般性，我们可以把任意字符串都看作某种图灵机)。显然，f_{uni}可以在多项式时间内被使用单步计数器的通用机器计算出来。为了证明 f_{uni}是一个弱单向函数(假如单向函数存在)，我们使用"可约性理论"。

假设单向函数存在，利用前面的结论，可以证明存在一个可以在二次时间内计算出来的单向函数 g。令 M_g 为计算 g 的二次时间机器。显然，一个(有效的)对 f_{uni}求逆的算法(当输入 $f_{\text{uni}}(\text{desc}(M_g),U_n)$时，此算法能以概率 $p(n)$对 f_{uni}求逆)可以容易地被转化为一个(有效的)对 g 求逆的算法(当输入 $g(U_n)$时该算法能以概率 $p(n)$对 g 求逆)。和命题 2.3.1 证明类似，可以证明对输入长为 $|\text{desc}(M_g)|+n$ 的字符串至少能以概率 $1-\epsilon(n)$对 f_{uni}求逆的算法可以产生一个对输入长为 n 的字符串至少能以概率 $1-2^{|\text{desc}(M_g)|}\cdot\epsilon(n)$对 g 求逆的算法(注意：$|\text{desc}(M_g)|$是个常数，只与 g 有关)。因此，如果 f_{uni}不是弱单向函数(即函数 ϵ 是可忽略的)，那么 g 也一定不是(弱)单向函数(即 $2^{|\text{desc}(M_g)|}\cdot\epsilon$ 也是可忽略的)。

根据定理 2.3.2(可以将弱单向函数 f_{uni}转化为强单向函数)，命题成立。

讨论：可以在特殊时间的界定内评估的单向函数对构造 f_{uni}来说是至关重要的，因为我们不可能构造一个对所有的多项式时间机器类型普遍适用的多项式时间机器(即可以"仿真"所有多项式时间机器的多项式时间机器)。然而，对每个多项式 $P(\cdot)$，构造这样一个多项式时间机器是可能的，该机器对所有在 $P(\cdot)$限定的时间内运行的机器都适用。

构造 f_{uni}的不切实际性源于 f_{uni}对巨长的输入(即其长度允许用非平常集的单向函数评估算法进行编码)很难求逆。而且，若要从 f_{uni}得到一个强单向函数，我们必须把后者作用于一个有大于 2^L 个输入的序列上，其中每个输入长度为 $L+n$，L 是潜在的单向函数编码长度的下界，n 是我们实际应用中的安全参数。

命题 2.4.1 还说明，原则上可以把单向函数是否存在这一问题，"转化"为判定一个特定的

函数是否是单向函数的问题。

2.4.2 单向函数类

前面给出的单向函数的描述方式只适合于抽象的讨论．如果要描述有许多固有的候选形式的单向函数，下面给出的方法（尽管更麻烦）可能更有用。我们不把单向函数视为在无穷域（即$\{0,1\}^*$）上操作的函数，而是视为在有限域上操作的函数的无穷集。这一集合中的函数共享一个评估算法，当输入一个函数的简洁描述及其作用域内的一个元素时，此算法返回这一特殊函数在给定点的值。函数类的简单描述对描述陷门置换和无爪函数也是非常有用的（分别参见 2.4.4 和 2.4.5 节）。下面我们从如下定义开始论述。

定义 2.4.2（函数类）：函数类是由指标组成的无限集（记为$\bar{I}$）及其对应的有限函数集（记为$\{f_i\}_{i\in\bar{I}}$）组成的。也就是说，对每个$i\in\bar{I}$，函数f_i的值域（记为D_i）是一个有限集。

特别地，指标集$\bar{I}$是所有字符串集合的“密集”子集；也就是说，n比特长的字符串在$\bar{I}$中所占的比例是不可忽略的（即$|\bar{I}\cap\{0,1\}^n|\geq 2^n/\mathrm{poly}(n)$）。

我们只对能应用于密码学的函数类感兴趣。前文已经暗示，使用函数类的必要条件是存在一个有效的函数评价算法（记为F），当输入$i\in\bar{I}$，$x\in D_i$时，此算法返回$f_i(x)$。然而此条件本身并不充分，我们必须既能在足够大的域上（随机）选择一个代表某一函数的指标，又能（在给定这个域的指标时）（随机）选取这个域上的一个元素。这个指标集的采样性质可以利用一个有效的算法（记为I）得到，当输入一个整数n（用一元表示）时，此算法随机选择一个代表某一函数的长为 poly(n)比特的指标及其相关的域（通常情况下，为了同有效算法（该算法在他们输入长度的多项式时间内运行）的标准相一致，我们使用一元表示法）。这个域的采样性质可以利用一个有效算法D得到，当输入一个指标i时，此算法随机选取D_i中的一个元素。这个函数类的单向性要求每个有效算法对给定的函数指标及其值域内的一个元素能对此函数求逆的概率是可以忽略的。此概率分布可以通过采样算法I和D得到。上文所述可归结为以下定义。

定义 2.4.3（单向函数类）：如果存在满足以下两个条件的 3 个概率多项式算法I，D和F，那么函数类$\{f_i: D_i\rightarrow\{0,1\}^*\}_{i\in\bar{I}}$被称作强单向函数类（弱单向函数类的定义类似）。

1. 易于采样和计算：

当输入1^n时，算法I的输出分布是集合$\bar{I}\cap\{0,1\}^n$中任意可变的指定值。当输入$i\in\bar{I}$时，算法D的输出分布是D_i中任意可变的指定值。当输入$i\in\bar{I}$和$x\in D_i$时，算法F总是输出$f_i(x)$。

（由此可得，$D_i\subseteq U_{m\leq\mathrm{poly}(|i|)}\{0,1\}^m$。为了不失一般性，假设$D_i\subseteq\{0,1\}^{\mathrm{poly}|i|}$，同样，为了不失一般性，算法$F$是确定性的。）

2. 难于求逆（适用于强单向函数）：对每个概率多项式时间算法A'、每个正多项式以及所有足够大的n，有

$$\Pr[A'(I_n,f_{I_n}(X_n))\in f_{I_n}^{-1}(f_{I_n}(X_n))]<\frac{1}{p(n)}$$

其中I_n是一个随机变量，表示输入1^n时算法I的输出分布；X_n也是一个随机变量，表示输入随机变量I_n时算法D的输出分布。

(适用于弱单向函数的难于求逆条件类似可得。)

需要强调的是,当输入 1^n 时算法 I 的输出不一定满足在 $\bar{I}\cap\{0,1\}^n$ 上均匀分布,甚至不要求 $I(1^n)$完全集中于单个字符串上。类似地,当输入 i 时算法 D 的输出不一定满足在 D_i 上均匀分布。但是难于求逆的条件隐含了 $D(i)$不能只集中于(关于 $|i|$ 的)多项式个字符串。还需要强调的是,对于算法 I 和 D 产生的分布状态(通常依赖于函数本身的映射),函数类难于求逆。

通过说明对应的 3 个子算法,我们可以描述一个单向函数类。然而,只有 3 个概率多项式时间子算法(I,D,F)都满足上述两个条件的函数类存在时,我们才可以说这些算法的 3 个概率多项式时间子算法(I,D,F)构成一个单向函数类。

显然,任何一个单向函数类可以表示为一个单向函数,反之亦然(参见习题 18),但是每种描述方式都有各自的优点。因此,为了表示常用的单向函数可选方式,我们将使用单向函数类这一描述方式。

宽松条件 为了减少单向函数类的固有候选形式的表达方式的复杂性,我们从两方面放宽了定义 2.4.3。首先,当输入 1^n 时允许算法 I 输出的指标长为 $p(n)$而不是 n,其中 $p(\cdot)$是一个多项式。其次,允许所有算法以可忽略的概率失效。最重要的是,只要 $I(1^n)\notin\bar{I}\cap\{0,1\}^{p(n)}$的概率是关于 n 的可忽略不计的函数,那么就允许指标采样算法 I 输出的字符串不在 $\bar{I}$ 中。(讨论时陷门置换和无爪函数可使用相同的宽松条件。)

附加性质:可有效识别的指标和域 对随后几个小节中即将讨论的某些单向函数类来说,有几个附加性质成立,这里我们只提及其中两个(有用的)附加性质:(1)含有可有效识别的指标集;(2)含有可有效识别的域集。也就是说,存在一个可判决 $\bar{I}$ 中成员的有效算法和一个对给定的 $i\in\bar{I}$ 和 x 可判决 $x\in D_i$ 是否成立的有效算法。需要注意的是,对没有放宽条件的定义 2.4.3 来说,用于产生 $i\in\bar{I}$(类似地有 $x\in D_i$)的硬币组成了对应的论题的认证(即 NP 见证);此 $i\in\bar{I}$(类似地有 $x\in D_i$)认证有助于对函数 f_i(类似地,有求 x 原像的函数)求逆。

2.4.3 单向函数类的实例

本节我们介绍几种基于数值计算理论的流行的单向函数类(例如,RSA 和离散对数)[①]。下面,我们要使用一些基本的数论知识和一些简单的数论算法。在附录 A 中我们将进一步讨论相关的数论资料。

2.4.3.1 RSA 函数

RSA 函数类包含一个由序对(N,e)组成的指标集,其中 N 是两个长为($\frac{1}{2}\cdot\log_2 N$)比特的素数(记为 P 和 Q)的乘积,e 是一个小于 N 的整数且与$(P-1)\cdot(Q-1)$互素。指标(N,e)的函数的定义域是$\{1,\ldots,N\}$,并将此域上的元素 x 映射到 $x^e \bmod N$ 上。根据 e 与$(P-1)\cdot(Q-1)$互素,可以证明此函数实际上是在此域上的一个置换。因此,RSA 函数类是一个置换函数类。

---

① 很明显这些仅仅是单向函数类的某种可选形式;它们的难于求逆性或是(广泛认同的)猜想或缘于(广泛认同的)猜想。

首先证明 RSA 函数类满足单向函数类的定义中给出的第一个条件(即易于采样和计算)。然后,我们给出 3 个子算法($I_{RSA}, D_{RSA}, F_{RSA}$)。

当输入 1^n 时,算法 I_{RSA} 均匀地选择两个素数 P 和 Q 以及一个整数 e,使得 $2^{n-1} \leq P < Q < 2^n$,e 与 $(P-1)\cdot(Q-1)$ 互素(特别地,e 均在可接纳的概率内选择)[①]算法 I_{RSA} 以输出 (N,e) 结束,其中 $N = P \cdot Q$。要高效率地实现 I_{RSA},我们需要一个概率多项式时间算法用来产生均匀(或几乎均匀)分布的素数。有关均匀素数发生器的更多细节,参见附录 A。

对算法 D_{RSA} 来说,当输入 (N,e) 时,它在集合 $D_{N,e} \stackrel{\text{def}}{=} \{1,\ldots,N\}$ 中(几乎均匀地)选择一个元素。(以指数趋于零是因为我们在无偏硬币投掷序列上进行 N 向选择的。)当输入 $((N,e),x)$ 时,F_{RSA} 的输出是

$$\mathrm{RSA}_{N,e}(x) \stackrel{\text{def}}{=} x^e \bmod N \tag{2.10}$$

对 N 进行因数分解是否可以转化为对 $\mathrm{RSA}_{N,e}$ 求逆还不可知,实际上这是一个众所周知的未决问题。但是已知的最好的对 $\mathrm{RSA}_{N,e}$ 求逆的算法就是(显式或隐式地)对 N 进行因数分解。无论如何,RSA 函数类难于求逆是被广泛认可的。

如前所述,$D_{N,e}$ 对应于模 N 加法群(因此包含 N 个元素)。另一方面,域 $D_{N,e}$ 必须是模 N 乘法群的元素(因此包含 $(P-1)\cdot(Q-1) \approx N - 2\sqrt{N} \approx N$ 个元素)。修正的域取样器在 $\{1,\ldots,N\}$ 选择一个元素,以避免所选的元素不与 N 互素这一不可靠的情况发生。前面定义的 $\mathrm{RSA}_{N,e}$ 函数引入了一个乘法群模 N 上的置换,所得的函数类和原函数类一样难于求逆(这个命题在习题中留给读者证明)。哪种描述方法更好似乎是一个个人喜好问题。

2.4.3.2 Rabin 函数

Rabin 函数类的定义与 RSA 函数类的定义类似,只是此函数是平方模 N(而不是 e 次幂模 N)。即

$$\mathrm{Rabin}_N(x) \stackrel{\text{def}}{=} x^2 \bmod N \tag{2.11}$$

然而此函数不能引入一个模 N 乘法群上的置换,而只能在这个群上进行 4 对 1 的映射。

可以证明,解出模 N 平方根在计算复杂度上等价于对 N 进行因数分解(即,在概率多项式时间变形上二者是可以互相转化的)。细节参见习题 21。因此,平方模一个合数是单向函数类当且仅当因数分解是困难的。需要提醒读者注意的是,一般来说整数的因数分解是困难的,当这一整数是两个等长的素数的乘积时此结论也成立。[②]

2.4.3.3 因数分解置换

对某一特殊的整数的子集(被称为 *Blum* 整数集)来说,前面定义的 $\mathrm{Rabin}_N(\cdot)$ 函数引入了一个模 N 二次剩余类上的置换。如果存在一个整数 x 使得 $r \equiv x^2 \pmod N$,那么 r 就是一个模 N 二次剩余。我们把模 N 乘法群上的二次剩余类记为 Q_N。如果 N 是两个与 3mod4 同余的素数的乘积,那么我们就说 N 是一个 *Blum* 整数。可以证明,当 N 是一个 *Blum* 整数时,Q_N 中每个元素都有一个惟一的平方根,此平方根也属于 Q_N。这就导致了函数类 $\mathrm{SQR} \stackrel{\text{def}}{=} (I_{BI},$

---

① 在某些参考资料中,令 e 等于 3。这时,应当选择这样两个素数(P 和 Q),使得它们等价于 2 mod 3。还不知道是否有如下假设:一个变量是单向的暗示着另一个也是单向的。

② 其实后一种情况被认为是最困难的。

D_{QR}, F_{SQR})置换的引入。当输入 1^n 时，算法 I_{BI} 均匀地选择两个素数 P 和 Q，使得 $2^{n-1} \leq P < Q < 2^n$，且 $P \equiv Q \equiv 3 \pmod 4$，然后输出 $N = P \cdot Q$。当输入 N 时，算法 D_{QR} 通过在模 N 乘法群上均匀地选择一个元素并求其模 N 平方来均匀地选择一个 Q_N 上的元素。算法 F_{SQR} 和 Rabin 函数类一样定义即可。这样，如果因数分解是困难的，所得的函数集就是单向的。

2.4.3.4 离散对数

另一类被广泛认为是困难的可计算的数论问题是在有限域(特别是素数基的有限域)中求解离散对数。DLP 函数类(其名字和对其单向性的猜想来自于离散对数问题)是通过 3 个子算法(I_{DLP}, D_{DLP}, F_{DLP})来定义的。

当输入 1^n 时，算法 I_{DLP} 均匀地选择一个素数 P，使得 $2^{n-1} \leq P < 2^n$，还有一个模 P 乘法群中的本原元素 G(即这个循环群中的一个生成元)，然后输出(P,G)。存在一个可以均匀地产生素数的概率多项式时间算法，并可以对 $P-1$ 进行因数分解，其中 P 是产生的素数(参见附录 A)。另一种方法是均匀地产生一个形为 $2Q+1$ 的素数 P，其中 Q 也是一个素数。然而，后一种方法必须假设关于此类素数的 DLP 是难题。此类素数一般是最困难的 DLP。利用对 $P-1$ 的因数分解，并通过在这个群中随机选择一个元素并校验其是否是 $P-1$ 阶(顺次求该元素的各阶幂，直至 $P-1$ 阶，并比较二者的结果是否等于 1)，我们可以找到一个本原元素。

对于算法 D_{DLP}，当输入(P,G)时，它均匀地选择一个以 $P-1$ 为模的余项。输入((P,G),x)时，算法 F_{DLP} 输出：

$$\mathrm{DLP}_{P,G}(x) \stackrel{\mathrm{def}}{=} G^x \bmod P \qquad (2.12)$$

因此，对 $\mathrm{DLP}_{P,G}$ 求逆等于对以 P 为模的离散对数(基为 G)求解。对每个上述形式的每一个(P,G)，函数 $\mathrm{DLP}_{P,G}$ 引入了一个从模 $P-1$ 加法群到模 P 乘法群的一一对应映射。因此，$\mathrm{DLP}_{P,G}$ 引入了一个集合$\{1,\ldots,P-1\}$上的置换。

其他群上的指数也可作为单向函数合法的候选形式，只要这个群上的离散对数问题是困难的。例如，在椭圆曲线上的点构成的群上，离散对数问题就是困难的。

2.4.4 陷门单向置换

下面我们将定义陷门(单向)置换并回忆一下流行的候选形式(即 RSA 函数类)。

2.4.4.1 定义

把单向函数类这一描述方式作为定义陷门置换的起始点是很方便的。简单地说，陷门置换就是一些具有附加属性的单向置换类$\{f_i\}$，f_i 一旦给定一个辅助输入作为指标 i 的陷门，就可以有效地对其求逆。指标 i 的陷门记为 $t(i)$，不能由 i 有效地求得，但是能有效地产生相应的序对(i,$t(i)$)。

定义 2.4.4(陷门置换类)：设 $I: 1^* \to \{0,1\}^* \times \{0,1\}^*$ 是一个概率算法，$I(1^n)$ 输出的第一个元素记为 $I_1(1^n)$。若满足以下两个条件，则 3 个算法(I,D,F)被称为强陷门置换类(弱陷门置换类的定义类似)。

1. 算法引入一个单向置换类：3 个算法(I_1,D,F)组成一个强单向置换类(类似地有弱单向置换类)。

（特别地，$F(i,x)=f_i(x)$。）

2. 根据陷门易于求逆：存在这样一个（确定性的）多项式时间算法，记为 F^{-1}，此算法对每个在 I 范围中的序对 (i,t) 和每个 $x\in D_i$，有 $F^{-1}(t,f_i(x))=x$。

若放宽这两个条件，则要求能以压倒多数的高概率满足这两个条件。即，允许产生指标的算法 I 以可忽略的概率输出序对 (i,t)，对此序对来说，或者 f_i 不是置换，或者对所有 $x\in D_i$，$F^{-1}(t,f_i(x))=x$ 不成立。另一方面，域采样算法（即 D）必须能在对应的域上产生一个几乎均匀分布的函数。综合所有变动，可以得到以下形式，此形式和定义 2.4.4 是无法比较的。我们同时也提出一个更简单的描述方式，并介绍了一个非均匀单向函数。

定义 2.4.5（修正的陷门置换类）：令 $\bar{I}\subseteq\{0,1\}^*$ 且 $\bar{I}_n\stackrel{\text{def}}{=}\bar{I}\cap\{0,1\}^n$，指标在 $\bar{I}$ 中的置换类就是集合 $\{f_i:D_i\to D_i\}_{i\in\bar{I}}$，其中每个 f_i 都是一个在对应域 D_i 上的一一对应映射。如果存在 4 个满足以下 5 个条件的概率多项式时间算法 I,D,F 和 F^{-1}，那么此函数类被称为陷门置换。

1. 指标和陷门的选择：对每个 n，有

$$\Pr[I(1^n)\in\bar{I}_n\times\{0,1\}^*]>1-2^{-n}$$

2. 域中选择：对每个 $n\in\mathbb{N}$ 和 $i\in\bar{I}_n$ 有，

(a) $\Pr[D(i)\in D_i]1-2^{-n}$；

(b) 若 $D(i)\in D_i$，则输出在 D_i 上均匀分布。也就是说，对每个 $x\in D_i$ 有

$$\Pr[D(i)=x\mid D(i)\in D_i]=\frac{1}{|D_i|}$$

于是，$D_i\subseteq U_{m=\mathrm{poly}(|i|)}\{0,1\}^m$。为了不失一般性，$D_i\subseteq\{0,1\}^{\mathrm{poly}(|i|)}$。

3. 有效评估：对每个 $n\in\mathbb{N}$，$i\in\bar{I}_n$，和 $x\in D_i$，有

$$\Pr[F(i,x)=f_i(x)]>1-2^{-n}$$

4. 难于求逆：设 I_n 是描述输出 $I(1^n)$ 中第一个元素的分布状态的随机变量，且 $X_n\stackrel{\text{def}}{=}D(I_n)$。考虑下面两个形式：

标准/均匀复杂度形式：对于每个概率多项式时间算法 A′、每个正多项式 P(·)以及所有足够大的 n，有

$$\Pr[A'(I_n,f_{I_n}(X_n))=X_n]<\frac{1}{p(n)}$$

非均匀复杂度形式：对每个多项式大小的电路族 $\{C_n\}_{n\in\mathbb{N}}$、每个正多项式 $p(\cdot)$ 以及所有足够大的 n，有

$$\Pr[C_n(I_n,f_{I_n}(X_n))=X_n]<\frac{1}{p(n)}$$

5. 根据陷门求逆：对每个 $n\in\mathbb{N}$、在 $I(1^n)$ 范围内的任意序对 (i,t)（其中 $i\in\bar{I}n$）以及每个 $x\in D_i$，有

$$\Pr[F^{-1}(t,f_i(x))=x]>1-2^{-n}$$

需要说明的是，可使条件 2、3、5 中任意一个条件不成立的，以指数形式趋于零的指标可以从 $\bar{I}$ 中忽略（可以用第 1 个条件中允许的错误概率来说明）。通过在所有可能的 $x\in D_i$ 上取均匀分布的概率，可以放宽第 3 和第 5 个条件。

2.4.4.2　RSA（及其因数分解）陷门

前面介绍的 RSA 函数类经过简单的修正也能具有陷门属性。为此，我们要修改算法

I_{RSA}，使之既能输出指标(N,e)又能输出陷门(N,d)，其中 d 是以$(P-1)\cdot(Q-1)$为模的 e 的逆(请注意由于 e 与$(P-1)\cdot(Q-1)$互质，所以 d 有这样的逆)。求逆算法 F_{RSA}^{-1}等价于算法 F_{RSA}(即 $F_{RSA}((N,d),y)=y^d \bmod N$。读者不难证明：对每个模 N 乘法群中的 x，有

$$F_{RSA}^{-1}((N,d),F_{RSA}((N,e),x)) = x^{ed} \bmod N$$

确实等于 x。实际上，我们可以证明对每个 x(即使 x 不与 N 互素)都有 $x^{ed}\equiv x \pmod N$。

前面介绍的 Rabin 函数类也可以按相同的方式修正，使得人们能够有效地计算所有 4 个给定的(模 N)二次剩余数的平方根。此时的陷门就是 N 的素因数。解出以 N 的每个素因数为模的平方根并结合中国剩余定理的结论，就可以计算以 N 为模的平方根。解出以给定素数为模的平方根的有效算法是已知的(参见附录 A)。更进一步，当 P 与 3 mod 4 同余时，通过计算 x 的$\frac{P+1}{4}$次幂就可以计算 P(同时也要转化中间以 P 为模的结果)。而且，当 N 是 Blum 整数时，前面介绍的函数类 SQR 就构成了一个陷门置换类(当然必须假设因数分解是困难的)。

2.4.5* 无爪 (claw-free)函数

把单向函数类这一描述方式作为定义无爪函数对的起始点也是很方便的。

2.4.5.1 定义

简单说来，无爪函数是由一个函数对的集合组成的，这些函数对易于计算，而且每对函数的成员有相同的域值，但是要找域值元素和这些函数作用下的原像是不可行的。

定义 2.4.6 无爪函数类)：函数对集合是由一个无限指标集(记为 $\bar{I}$)、两个有限集 D_i^0 和 D_i^1(对每个 $i\in\bar{I}$)以及分别定义在 D_i^0 和 D_i^1 上的两个函数 f_i^0 和 f_i^1 组成的。如果存在 3 个满足以下条件的概率多项式时间算法 I,D,F，则称这样的函数类为无爪函数。

1. 易于取样和计算：设随机变量 $I(1^n)$是集合 $\bar{I}\cap\{0,1\}^n$ 中一个给定的值。对每个 $i\in\bar{I}$ 和 $\sigma\in\{0,1\}$，随机变量 $D(\sigma,i)$在 D_i^σ 上分布，且对每个 $x\in D_i^\sigma$，有 $F(\sigma,i,x)=f_i^\sigma(x)$。

2. 相同的域值分布：对索引集 $\bar{I}$ 中的每个 i，随机变量 $f_i^0(D(0,i))$和 $f_i^1(D(1,i))$满足相同的分布。

3. 难于构成爪子：满足 $f_i^0(x)=f_i^1(y)$的序对(x,y)被称作指标 i 的爪子。令 C_i 为指标 i 的爪子的集合，对于每个概率多项式时间算法 A'、每个正多项式 $p(\cdot)$和所有足够大的 n，有

$$\Pr[A'(I_n)\in C_{I_n}] < \frac{1}{p(n)}$$

其中 I_n 是一个描述输入 1^n 时算法 I 的输出分布状态的随机变量。

定义 2.4.6 中的第一个要求和定义 2.4.3 中的第一个要求类似。定义 2.4.6 的其他两个要求本质上是冲突的。一方面，它(至少)要求爪子的确存在，而另一方面又要求不能有效地找到爪子。显然，无爪函数类构成了强单向函数类(参见习题 22)。下面讨论一种特殊情况：当两个域相同时(即 $D_i\stackrel{\text{def}}{=}D_i^0=D_i^1$)，随机变量 $D(\sigma,i)$在 D_i 上均匀分布，而且函数 f_i^0 和 f_i^1 是 D_i 上的置换。这样一个函数类就是无爪置换对类。

此外，如果允许算法(即 I、D、F)以可忽略的概率失效，那么可以放宽定义 2.4.6 的条件。(无爪)函数类具有(或不具有)的附加属性是一个可被有效识别的指标集(即是一个可以判决

给定的字符串是否属于 $\overline{I}$ 的概率多项式时间算法)。

2.4.5.2 DLP 无爪函数类

下面我们着手证明在特殊、合理的困难性假设下,无爪函数类确实存在。首先我们假设以素数为基的离散对数问题(DLP)是困难的,并由此提出一个函数类。

以下是对(基于上述假设的)无爪置换对类的描述。指标集由三部分组成(P,G,Z),其中 P 是一个素数,G 是一个以 P 为模的本原元素,Z 是一个在以 P 为模的余项域上的元素。指标采样算法像 2.4.3 节提出的 DLP 函数集那样选择 P 和 G,在 P 为模的余项域上均匀选取 Z。两个以(P,G,Z)为指标的函数的域是相同的,等于集合$\{1,\ldots,P-1\}$,域采样算法则在此集合上均匀地选择。对函数本身,令

$$f^{\sigma}_{P,G,Z}(x) \stackrel{\text{def}}{=} Z^{\sigma} \cdot G^{x} \bmod P \tag{2.13}$$

读者不难证明这两个函数是$\{1,\ldots,P-1\}$上的置换。实际上,函数 $f^{0}_{P,G,Z}$和 2.4.3 小节提出的函数 $\mathrm{DLP}_{P,G}$是一致的。而且,构成指标(P,G,Z)的爪子要求找到一个以 G 为基以 P 为模的离散对数 Z(因为 $G^{x}\equiv Z\cdot G^{y}(\mathrm{mod} P)$,可推导出 $G^{x-y}\equiv Z(\mathrm{mod}\ P)$)。于是,对指标集不可忽略的部分求爪子的问题就转化成了对 2.4.3 小节提出的 DLP 函数类求逆的问题。换句话说,如果 DLP 函数类是单向的,那么等式 2.13 定义的置换对集合是无爪的。

前文所述的函数类不具备含有可有效识别的指标集这一附加属性,因为它不知道如何有效地识别以素数为模的本原元素。这点可以通过对 DLP 的困难性做更严格的假设来修正。特别地,我们假设即使作为附加输入给出乘法群大小的数可以因数分解,DLP 还是困难的。这一假设允许把 $P-1$ 的因数分解加到指标描述中去,从而使得指标集可被有效地识别(因为通过计算 G 的形如$(P-1)/Q$ 次幂,就可以判断 G 是否是本原元素)。如果 DLP 对形如 $2Q+1$ 的素数来说也是困难的,Q 是素数,判决就更容易了:只要简单计算一下 $G^{2} \bmod P$ 和 $G^{(P-1)/2} \bmod P$,并校验一下这两个数是否不同于 1,就可以判断 G 是否是一个以 P 为模的本原元素。

2.4.5.3 基于因数分解的无爪函数类

现在我们证明若假设整数因数分解是不可行的,那么无爪函数类的确存在。在下面的描述中,我们使用了 Blum 整数(即两个等价于 3 mod 4 的素数的乘积)的结构特性,Blum 整数将在附录 A 中进一步讨论。特别地,对一个 Blum 整数 N,有

- $-1 \bmod N$ 的雅可比符号(Jacobi symbol)等于 1;
- 在每个二次剩余类的平方根中有一半具有雅可比符号 1。

在含雅可比符号$+1$(或-1)的模 N 乘法群中的剩余类记为 J_N^{+1}(或 J_N^{-1})。

此函数类的指标集是由两个等长的素数合成的所有 Blum 整数组成的。当输入 1^n 时,指标选择算法通过均匀地选择两个等价于 3 mod 4 的(n 比特)素数,来均匀地选择这样一个整数,并输出它们的积,记为 N。尽管两个指标 N 的函数(分别记为 f_N^0 和 f_N^1)组成了模 N 平方,但是它们对应的作用域是不相关的。函数 f_N^{σ} 的作用域等于集合 $J_N^{(-1)^{\sigma}}$。域采样算法记为 D,按理想的方法在对应的域上均匀地选择元素。特别地,当输入(σ,N)时,算法 D 均匀地选择多项式个模 N 剩余类,并输出第一个具有雅可比符号$(-1)^{\sigma}$ 的剩余类。

读者不难证明,$f_N^0(D(0,N))$和 $f_N^1(D(1,N))$都在模 N 二次剩余类上均匀分布。一个爪

子产生两个余项 $x\in J_N^{+1}$ 和 $y\in J_N^{-1}$，它们的模 N 平方值相等（即 $x^2\equiv y^2 \pmod N$），根据这个事实可以证明生成爪子的困难性。由 $-1\in J_N^{+1}$（后者是一个乘法子群）可得 $y\not\equiv\pm x \pmod N$，因此 y＋x 和 N 的最大公约数（g. c. d.）生成了 N 的因数分解。

上述函数类是由两个（定义在不相关域上的）2 对 1 的函数组成的。为了得到一个无爪置换类，我们按如下方法稍微修改一下函数类。指标集是由 Blum 整数组成的，Blum 整数是两个等长的素数 P 和 Q 的积，且 $P\equiv 3 \pmod 8$，$Q\equiv 7 \pmod 8$。对这样的合数来说，2 和 -2 都不是以 $N=P\cdot Q$ 为模的二次剩余项（实际上 $\pm 2\in J_N^{-1}$）。考虑定义在模 N 二次剩余类 Q_N 上的函数 f_N^0 和 f_N^1，有

$$f_N^{\sigma}(x) \stackrel{\text{def}}{=} 4^{\sigma}\cdot x^2 \bmod N \tag{2.14}$$

显然，f_N^0 和 f_N^1 都是在 Q_N 上的置换。一个爪子产生两个二次余项 x 和 y，满足 $x^2\equiv 4y^2 \pmod N$，根据这个事实可以证明生成爪子的困难性。于是，$(x/y)^2\equiv 4 \pmod N$，因此，$(2-(x/y))\cdot(2+(x/y))\equiv 0 \pmod N$。由 $\pm 2\notin Q_N$（后者是一个乘法子群），可得 $(x/y)\not\equiv\pm 2 \pmod N$，从而 $(2\pm x\cdot y^{-1} \bmod N)$ 和 N 的 g. c. d. 生成了 N 的因数分解。

我们不知道上述函数类具备含有可有效识别的指标集这一附加属性。特别地，我们甚至不知道如何有效地区分两个素数的乘积和多个素数的乘积。

2.4.6* 关于推荐候选式

虽然我们相信单向函数是存在的，但在实际应用中仅仅存在是不够的。典型地，基于单向函数的应用程序要求详细说明一个具体的（候选单向）函数[①]。因此，为单向函数提出合理的候选形式具有重要的实际意义。大家都知道这样一个合理的（单向函数）候选形式应当含有一个有效的函数评价算法。这里“函数”是以单向函数类的概念提出的，域采样器和函数评价算法应当是非常有效的（至于指标采样器，“中等效率”即可）。然而，人们似乎不太关心对他们提出的候选形式求逆的困难度。需要强调的是，候选形式在“大多数”情况（它与由候选函数决定的实例分布有关）下必须难于求逆，而不仅仅是在最坏的情况下。“大多数情况下的困难性”（这不同于最坏情况分析）非常容易受实例分布的影响。因此，人们十分关心如何从某一分布的大多数情况下的复杂度推导出另一分布的大多数情况下的复杂度。在这一研究领域的短暂的研究史中，这一点被忽略了。因此，做了一些坏的推测。

举例说来，可以考虑如下基于图的同构问题的困难性猜想而提出的（坏）基本单向函数。设 $F_{GI}(G,\pi)=(G,\pi G)$，其中 G 是一个无向图，π 是一个在其顶点集上的置换，πG 表示根据 π 重命名 G 的顶点而得到的图（即 $(\pi(u),\pi(v))$ 是 πG 的一条边当且仅当 (u,v) 是 G 的一条边）。尽管图的同构问题不能在多项式时间内解决是被大家认同的，但是不难看出在大多数实例中，F_{GI} 易于求逆（例如，可以通过统计顶点度数来判断是否同构）。也就是说，对最坏情况下的困难性的猜想不包含均匀分布时大多数情况下的困难性。而且，即使某一问题对某一分布状态在大多数情况下是困难的，也必须详细描述这一分布状态并提出一个有效的采样算法。

---

① 正如 2.4.1 小节中所说的，证明通用单向函数的存在性没有什么实际价值。

2.5 核心断言(Hard-Core Predicates)

简单地说,一个函数 f 是单向的就意味着给定一个 y,求它在 f 作用下的原像是不可行的。但这并不意味着求 f 作用下 y 的原像的部分信息也是不可行的。特别地,找回原像的一半比特位可能是很容易的(例如,给定一个单向函数 f,对每一个 $|x|=|r|$,考虑函数 $g(x,r) \stackrel{\text{def}}{=} (f(x),r)$)。单向函数不必隐藏部分信息这一事实限制了它在像安全加密这样的工作中的"直接应用性"。值得庆幸的是,如果假设单向函数存在,就可能构造能够隐藏原像的重要部分信息(从原像出发很容易计算)的单向函数。这部分信息被认为是对 f 难求逆的"核心"困难。

2.5.1 定义

简单地说,一个多项式时间断言 b 被称作是一个函数 f 的核心,如果对每一个有效的算法,给定 $f(x)$,能以略大于 1/2 的成功概率猜中 $b(x)$。

定义 2.5.1(核心断言):一个多项式时间可计算的断言 $b:\{0,1\}^* \to \{0,1\}$,如果对每一个概率多项式时间算法 A'、每一个正多项式 $p(\cdot)$ 和所有足够大的 n,都有

$$\Pr[A'(f(U_n)) = b(U_n)] < \frac{1}{2} + \frac{1}{p(n)}$$

则称 b 是函数 f 的核心。

注意,对每一个 $b:\{0,1\}^* \to \{0,1\}$ 和 $f:\{0,1\}^* \to \{0,1\}^*$ 存在明显的算法,至少能以 1/2 的成功概率由 $f(U_n)$ 推导出 $b(U_n)$(例如,一个算法如果遗忘了其输入,则输出位服从均匀分布)。同样,如果 b 是核心断言(对任意函数),那么 $b(U_n)$ 一定几乎是无偏的(即,$|\Pr[b(U_n)=0]-\Pr[b(U_n)=1]|$ 一定是 n 的一个可忽略的函数)。

既然 b 本身是多项式时间可计算的,从 $f(x)$ 估计 $b(x)$ 的有效算法失败(成功的不可忽略的概率高于 1/2)的原因要么是 f 信息的丢失(即 f 不是 1-1 的),要么是对 f 求逆的困难性。例如,断言 $b(\sigma\alpha)=\sigma$ 是函数 $f(\sigma\alpha) \stackrel{\text{def}}{=} 0\alpha$ 的核心,这里 $\sigma\in\{0,1\}$,$\alpha\in\{0,1\}^*$。因此,这种情况下,b 是函数 f 的核心是因为 f 丢失信息这一事实(特别是首位 σ)。另一方面,如果 f 没有丢失任何信息(即 f 是 1-1 的),此时只有当 f 是单向函数时(见习题 25),f 的核心才存在。我们感兴趣的是,由 $f(x)$ 估计 $b(x)$ 的难度不是由信息理论的原因造成的(即信息丢失),而是由计算方面的原因造成的。

对于单向函数类的核心断言可类似地定义。一般地,断言依赖于函数指数,同时这个指数也被传给两个算法(即估计和依据函数值断言的算法)。也就是说,如果对每一个概率多项式时间算法 A'、每一个正多项式 $p(\cdot)$ 和所有足够大的 n,都有

$$\Pr[A'(I_n, f_{I_n}(X_n)) = B(I_n, X_n)] < \frac{1}{2} + \frac{1}{p(n)}, \text{这里 } I_n \stackrel{\text{def}}{=} I(1^n), X_n \stackrel{\text{def}}{=} D(I_n)$$

则称多项式时间算法 $B:\{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ 是单向函数类 (I,D,F) 的核心。

一些自然的候选。 倘若相应的类是单向的,简单的核心断言都是以 RSA,Rabin 和 DLP 集合(2.4.3 节提到的)著称的,特别地,如果 RSA 集合是单向的,则它的最低有效位(the least

significant bit)就是它的核心。即，假设 RSA 集合是单向的，那么由 $RSA_{N,e}(x)=x^e \bmod N$ 来猜测(以远远大于 1/2 的成功概率)x 的最低有效位是不可行的。同样，假设整数分解是难处理的，那么从 $Rabin_N(x)=x^2 \bmod N$ 猜测 $x\in Q_N$ 的最低有效位是不可行的，这里 N 是一个 Blum 整数，(Q_N表示模 N 的二次剩余集合)。最后，假设 DLP 集合是单向的，当给定 $DLP_{P,G}(X)=G^x \bmod P$ 时，猜测是否有 $\chi<\frac{P}{2}$是不可行的。下一节我们将给出这一类型的一般结论。

2.5.2　任意单向函数的核心断言

实际上，标题是不正确的：我们只计划给出特定形式的(强)单向函数的核心断言。当然，在“安全”和“效率”都没有实质性丢失的情况下，每一个(强)单向函数都可以很容易地转化为要求形式的函数。

定理 2.5.2：设 f 是任意的强单向函数，定义 g 如下，$g(x,r)\stackrel{def}{=}(f(x),r)$，这里 $|x|=|r|$。令 $b(x,r)$表示二进制向量 x 和 r 的内积模 2，则断言 b 是函数 g 的核心。

换句话说，定理说明如果 f 是强单向的，给定 $f(x)$和它的子集时，猜测 x 的位的随机子集的异或(XOR)是不可行的。我们强调定理要求 f 是强单向的，如果 f 仅仅是弱单向的(见习题 25)，结论就是错误的。显然，g 也是强单向的。g 保持了 f 的其他性质，诸如等长度和 1-1 的。此外，对有/无陷门的单向函数类可得出类似的结论。

本节剩下的部分主要是证明定理 2.5.2。我们再用“可约性理论”：这里，对 f 求逆转化为由$(f(x),r)$猜测 $b(x,r)$。因此，我们假设(建立一个矛盾)存在一个有效的算法，能以不可忽略的概率来猜测内积，从而我们得到一个以相关的(即不可忽略的)成功概率对 f 求逆的算法。这与 f 是单向函数相矛盾。

我们以预备知识和对动机的讨论开始，从而引出证明的主要部分。最后以支持“更高安全性”的可约性理论的更有效实现总结本节。

2.5.2.1　预备知识

设 G 为一个(概率多项式时间)算法，当输入 $f(x)$和 r 时，试着猜出 x 和 r 的内积(模 2)。用 $\varepsilon_G(n)$表示算法 G 由 $f(x)$和 r 猜出 $b(x,r)$的概率，这里 x 和 r 都是在$\{0,1\}^n$ 上均匀选择的。即，

$$\varepsilon_G(n)\stackrel{def}{=}\Pr[G(f(X_n),R_n)=b(X_n,R_n)]-\frac{1}{2} \tag{2.15}$$

这里和以后，X_n和 R_n表示两个相互独立的均匀分布在$\{0,1\}^n$ 上的随机变量。假设 b 不是 g 的核心，这就意味着存在一个有效的算法 G、一个多项式 $p(\cdot)$和一个无限集 N，使得对每一个 $n\in N$，都有 $\varepsilon_G(n)>\frac{1}{p(n)}$。我们集中注意力考虑算法 G 和集合 N 中的 n，后面我们将 ε_G 简记为 ε。

第一个结论是当输入长度为 n 的 x 的至少$\frac{\varepsilon(n)}{2}$位时，算法 G 至少以$\frac{\varepsilon(n)}{2}$的概率由 $f(x)$和 R_n猜出 $b(x,R_n)$。

结论 2.5.2.1：存在一个集合 $S_n\subseteq\{0,1\}^n$，它的基至少为$\frac{\varepsilon(n)}{2}\cdot 2^n$，使得对每一个 $x\in S_n$，都有

$$s(x) \stackrel{\text{def}}{=} \Pr[G(f(x),R_n) = b(x,R_n)] \geq \frac{1}{2} + \frac{\epsilon(n)}{2}$$

这里的概率对 R_n 的所有可能值和算法 G 的所有内部掷币序列都是适用的，而 x 却是固定的。

证明：此结论是依据一般的证明方法来证明的，即根据 $\mathrm{E}(s(X_n)) = \frac{1}{2} + \epsilon(n)$，再应用马尔可夫不等式。

最后，我们注意 S_n 中的 x。应该提出一个有效的算法，对每一个输入 y，能够以很高的概率找到 x，满足 $y = f(x)$，且 $x \in S_n$。如果 $\Pr[U_n \in S_n] \geq \frac{\epsilon(n)}{2}$，就会得出与 f 是强单向函数相悖的矛盾。

我们以一个诱导性的讨论开始。算法 G 作为子程序的求逆算法，将在后面正式描述并分析。

2.5.2.2　一个诱导性的讨论

考虑固定的 $x \in S_n$。按照定义，$s(x) \geq \frac{1}{2} + \frac{\epsilon(n)}{2} > \frac{1}{2} + \frac{1}{2p(n)}$。暂时假设 $s(x) > \frac{3}{4} + \frac{1}{2p(n)}$。当然，我们没有理由相信这样的事实，我们仅仅是在做一个逻辑假设。并且，在这种情况下（即 $s(x) > \frac{3}{4} + \frac{1}{\mathrm{poly}(|x|)}$），由 $f(x)$ 求 x 非常容易。为了得到 x 的第 i 个位，记为 x_i，我们随机选择 $r \in \{0,1\}^n$，并计算 $G(f(x),r)$ 和 $G(f(x),r \oplus e^i)$，这里 e^i 是一个 n 维二进制向量，第 i 位是 1，其他位都是 0，$v \oplus u$ 表示二进制向量 u 和 v 的模 2 相加和。（上述过程通过独立地随机选择 r，重复多项式次，从而以多数为准的原则来确定 x_i。）

如果 $G(f(x),r) = b(x,r)$，且 $G(f(x),r \oplus e^i) = b(x,r \oplus e^i)$，则

$$G(f(x),r) \oplus G(f(x),r \oplus e^i) = b(x,r) \oplus b(x,r \oplus e^i) = b(x,e^i) = x_i$$

这里第二个等式利用了

$$b(x,r) \oplus b(x,s) \equiv \sum_{i=1}^{n} x_i r_i + \sum_{i=1}^{n} x_i s_i \equiv \sum_{i=1}^{n} x_i (r_i + s_i) \equiv b(x, r \oplus s) \pmod 2$$

对一个随机数 r，$G(f(x),r) = b(x,r)$ 和 $G(f(x),r \oplus e^i) = b(x,r \oplus e^i)$ 成立的概率至少为 $1 - 2 \cdot (\frac{1}{4} - \frac{1}{\mathrm{poly}(|x|)}) > \frac{1}{2} + \frac{1}{\mathrm{poly}(|x|)}$。因此，重复上述步骤足够多的次数，并遵循多数原则，就能以很高的概率获得 x_i。类似地，我们可以获得 x 的全部位，从而完成了已知 $f(x)$ 求 x 这一求逆问题。然而，我们全部的分析都是在假设（并没有得到证实）$s(x) > \frac{3}{4} + \frac{1}{2p(|x|)}$ 的情况下进行的，而实际上我们只知道 $s(x) > \frac{1}{2} + \frac{1}{2p(|x|)}$。

上述过程的问题是把算法 G 在输入形式为 $(f(x), \cdot)$ 时的错误概率加倍了。如果假设 G 的一般错误概率是小于 $\frac{1}{4}$ 的，且为不可忽略的，在这种不现实的假设下，将错误概率加倍不会引起任何问题。然而一般情况下（即使在特殊情况下，G 的错误概率恰好等于 $\frac{1}{4}$），前面的过程是很难对 f 求逆的。注意，G 的平均错误概率（对所有可能的形式输入 $(f(x), \cdot)$ 求平均）并不能因为重复 G 多次而降低（例如，G 可能总是在 $\frac{3}{4}$ 的输入时正确，而在其余 $\frac{1}{4}$ 的输入时错误）。这就要求我们找应用算法 G 的另一种方法，不会将算法 G 的原始错误概率加倍。关键是找一种产生 r 的方法，对每一个 r（和 i）只用算法 G 一次，而不是两次。特别地，我们必须用不同的方法获得对 $b(x,r)$ 以及 $b(x,r \oplus e^i)$ 的"猜测"。好的一面是因为只用 G 来"猜测" $b(x,r \oplus e^i)$，错误概率就不会加倍；坏的一面是我们仍需要知道 $b(x,r)$，而不知道不用 G 如何知道

$b(x,r)$。那么我们只能自己猜测 $b(x,r)$。这对于我们只猜测对应于一个 r 的 $b(x,r)$（或 $|x|$ 的对数多个 r）或许可行，但问题是我们必须知道对应于多项式多的 r 的 $b(x,r)$ 值。用一些明显的猜测 $b(x,r)$ 的方法都会导致呈指数下降的成功概率。如果我们能够生成这些多项式许多的 r，使得一方面它们是“足够随机的”，而另一方面，我们能以可注意到的概率猜测所有 $b(x,r)$ 的值。特别地，用两两独立的方式生成 r 就可以满足上述两个（看似互相矛盾）的要求。以前强调过，当能够成功猜测所有 $b(x,r)$ 的值时，就能以很高的概率获得 x。因此，我们能以可注意的概率获得 x。

在生成两两独立的 r 的过程中（可猜出相应的 $b(x,r)$ 的值），需注意的是顺序。为了生成 $m=\text{poly}(n)$ 多的 r，我们在 $\{0,1\}^n$ 中均匀（独立）地选择 $l \stackrel{\text{def}}{=} \log_2(m+1)$ 串。记这些串为 $s^1,\ldots,s^l$，则我们可通过 $b(x,s^i)$ 猜测 $b(x,s^1)$。记这些从 $\{0,1\}$ 中均匀（独立）选择的猜测为 $\sigma^1\ldots\sigma^l$。这样我们对所有猜测的 $b(x,s^i)$ 正确概率为 $2^{-l}=\frac{1}{\text{poly}(n)}$。不同的 r 对应不同 $\{1,2,\ldots,l\}$ 的非空子集。特别地，设 $r^J \stackrel{\text{def}}{=} \bigoplus_{j\in J} s^j$，读者很容易就能够证明 r^J 是两两相互独立的，每一个都服从 $\{0,1\}^n$ 上的均匀分布。关键是观察

$$b(x,r^J) = b(x,\bigoplus_{j\in J}s^j) = \bigoplus_{j\in J}b(x,s^j)$$

因此，我们对 $b(x,r^J)$ 的猜测是 $\bigoplus_{j\in J}\sigma^j$，并以可注意的概率保证猜测是正确的。

2.5.2.3 返回到实际证明

下面是对求逆算法的正式描述，记为 A。简单一些，我们假设 f 是等长度的（但这个假设不是本质的）。当输入 y 时（假设在 f 的定义域内），算法 A 使得 $n \stackrel{\text{def}}{=} |y|$，$l \stackrel{\text{def}}{=} \lceil \log_2(2n\cdot p(n)^2+1)\rceil$，这里 $p(\cdot)$ 是前面声明的多项式（即，对无限多的 $n\in N$，有 $\varepsilon(n)>\frac{1}{p(n)}$）。算法 A 的步骤如下：

1. 均匀独立地选择 $s^1,\ldots,s^l\in\{0,1\}^n$，$\sigma^1,\ldots\sigma^l\in\{0,1\}$。
2. 对每一个非空子集 $J\subseteq\{1,2,\ldots,l\}$，计算一个串 $r^J\leftarrow\bigoplus_{j\in J}s^j$ 和一个位 $\rho^J\leftarrow\bigoplus_{j\in J}\sigma^j$。
3. 对每一个 $i\in\{1,\ldots,n\}$ 和每一个非空子集 $J\subseteq\{1,\ldots,l\}$，计算 $z_i^J\leftarrow\rho^J\oplus G(y,r^J\oplus e^i)$。
4. 对每一个 $i\in\{1,\ldots,n\}$，令 z_i 的值为 z_i^J 所有取值的半数以上。
5. 输出 $z=z_1,\ldots z_n$。

备注：另一种可选择的实现方案。在这种方案中，求逆算法尝试 $\sigma^1,\ldots\sigma^l$ 所有可能的取值，对 2^l 中每一个可能值计算一个串 z，而只输出一个最满足 $f(z)=y$ 的串 z，为今后参考方便，记这种可选择方案为 A'（下一节将做进一步讨论）。

下面是对每一个 $x\in S_n$，当输入形式为 $f(x)$ 时算法 A 成功概率的详细分析，这里 $n\in N$。广泛应用的一个关键需注意的问题是对 $x,\alpha,\beta\in\{0,1\}^n$，有

$$b(x,\alpha\oplus\beta) = b(x,\alpha)\oplus b(x,\beta)$$

它是由 $b(x,r^J)=b(x,\bigoplus_{j\in J}s^j)=\bigoplus_{j\in J}b(x,s^j)$ 得来的。分析的主要部分指出，在 σ^j 正确的情况下（即，对所有的 $j\in\{1,\cdots,l\}$，都有 $\sigma^j=b(x,s^j)$），对所有的 $i\in\{1,\cdots,n\}$，$z_i=x_i$ 以固定的概率成立。这一点通过限定 z_i^J 的大多数等于 x_i 的概率小于某一界的前提来证明，这里 $z_i^J=b(x,r^J)\oplus G(f(x),r^J\oplus e^i)$ 来源于假设：对所有 $j\in\{1,\cdots,l\}$，都有 $\sigma^j=b(x,s^j)$。

结论 2.5.2.2：对每一个 $x\in S_n$ 和每一个 $1\le i\le n$，

$$\Pr\left[\,|\{J:b(x,r^J)\oplus G(f(x),r^J\oplus e^i)=x_i\}|>\frac{1}{2}\cdot(2^l-1)\right]>1-\frac{1}{2n}$$

这里，$r^J \stackrel{\text{def}}{=} \bigoplus_{j\in J} s^j$，且每一个 s^j 是相互独立均匀地在 $\{0,1\}^n$ 上选择的。

证明：对每一个 J，定义一个 0-1 随机变量 ζ^J，使得 $\zeta^J=1$ 当且仅当 $b(x,r^J)\oplus G(f(x),r^J\oplus e^i)=x_i$。既然 $b(x,r^J)\oplus b(x,r^J\oplus e^i)=x_i$，则 $\zeta^J=1$ 当且仅当 $G(f(x),r^J\oplus e^i)=b(x,r^J\oplus e^i)$。

读者很容易验证每一个 r^J 服从 $\{0,1\}^n$ 上的均匀分布，并且每一个 $r^J\oplus e^i$ 也服从相同分布，从而可得出每一个 ζ^J 以概率 $s(x)$ 等于 1。当 $x\in s_n$ 时，$s(x)$ 至少为 $\frac{1}{2}+\frac{1}{2p(n)}$。由 r^J 是两两独立的可推测 ζ^J 是两两独立的。为了不失一般性，对每一个 $J\neq K$，存在 $j\in J, k\in K-J$. 因此，对每一个 $\alpha,\beta\in\{0,1\}^n$，我们有

$$\begin{aligned}\Pr[r^K=\beta\,|\,r^J=\alpha] &= \Pr[s^k=\beta\,|\,s^j=\alpha] \\ &= \Pr[s^k=\beta] \\ &= \Pr[r^K=\beta]\end{aligned}$$

从而可得出 r^J 是两两独立的。令 $m\stackrel{\text{def}}{=}2^l-1$，$\zeta$ 代表一类 ζ^J（都服从相同分布）。应用 Chebyshev 不等式（和 $m\geq 2n\cdot p(n)^2$），我们得到

$$\begin{aligned}\Pr\Big[\sum_J \zeta^J \leq \frac{1}{2}\cdot m\Big] &\leq \Pr\Big[\Big|\sum_J \zeta^J-\Big(\frac{1}{2}+\frac{1}{2p(n)}\Big)\cdot m\Big|\geq \frac{1}{2p(n)}\cdot m\Big] \\ &\leq \frac{m\cdot \mathrm{Var}[\zeta]}{\Big(\frac{1}{2p(n)}\cdot m\Big)^2} \\ &= \frac{\mathrm{Var}[\zeta]}{\Big(\frac{1}{2p(n)}\Big)^2\cdot(2n\cdot p(n)^2)} \\ &< \frac{\frac{1}{4}}{\Big(\frac{1}{2p(n)}\Big)^2\cdot(2n\cdot p(n)^2)} \\ &= \frac{1}{2n}\end{aligned}$$

结论得证。□

如果对所有的 j，有 $\sigma^j=b(x,s^j)$，则对所有非空的 J，有：$\rho^J=\bigoplus_{j\in J}\sigma^j=\bigoplus_{j\in J}b(x,s^j)=b(x,r^J)$。在此情况下，算法 A 至少以 $\frac{1}{2}$ 的概率输出串 $z=x$。然而第一个事件（即，对所有的 j，有 $\sigma^j=b(x,s^j)$）与结论 2.5.2.2 所分析的事件相互独立，并以 $2^{-l}=\frac{1}{2n\cdot p(n)^2+1}$ 的概率发生。因此 $x\in s_n$ 时算法 A 对 f（输入 $f(x)$）求逆的概率至少为 $\frac{1}{2}\cdot 2^{-l}=\frac{1}{4n\cdot p(|x|)^2+2}$（而选择算法 A' 的成功概率至少为 $\frac{1}{2}$）。利用（结论 2.5.2.1）$|S_n|>\frac{1}{2p(n)}\cdot 2^n$，我们可推导出对每一个 $n\in N$，算法 A 对 f 求 $f(U_n)$ 原像的成功概率至少为 $\frac{1}{8n\cdot p(n)^3+4p(n)}$。注意到 A 是多项式时间算法（即，除去进行多项式数量的其他计算外，仅仅调用 G $2n\cdot p(n)^2=\mathrm{poly}(n)$ 次），就可以得出与我们假设 f 是强单向函数相矛盾的结论。■

2.5.2.4* 一些更有效的变形

前述证明实际上建立了如下命题。

命题 2.5.3：令 G 为一个运行时间为 $t_G:N\to N$ 的概率算法，猜测 b（见等式 2.15 的成功概

率为 $\varepsilon_G: N \to [0,1]$. 那么就存在算法 A 运行时间为 $O(n^2/\varepsilon_G(n)^2)\cdot t_G(n)$，使得

$$\Pr[A(f(U_n)) = U_n] \geq \frac{\varepsilon_G(n)}{2}\cdot\frac{\varepsilon_G(n)^2}{4n}$$

前面提到的选择方案 A'（即试验 σ^i 的所有可能值，而不是猜测其中的某一个），运行时间为 $O(n^3/\varepsilon_G(n)^4)\cdot t_G(n)$，并满足

$$\Pr[A'(f(U_n)) = U_n] \geq \frac{\varepsilon_G(n)}{2}\cdot\frac{1}{2}$$

下面，我们提出一个更有效的 A' 的实施方案。把它与更精确的平均理论而不仅仅是与结论 2.5.2.1 中用到的理论相结合，我们得到如下命题。

命题 2.5.4：设 $G, t_G: N \to N$ 和 $\varepsilon_G: N \to [0,1]$ 如前，定义 $l(n) \stackrel{\text{def}}{=} \log_2(1/\varepsilon_G(n))$。那么存在算法 A''，其运行时间如预期的那样，为 $O(n^2\cdot l(n)^3)\cdot t_G(n)$，并满足

$$\Pr[A''(f(U_n)) = U_n] = \Omega(\varepsilon_G(n)^2)$$

因此 A'' 的时间与成功率之比为 $\text{poly}(n)/\varepsilon_G(n)^2$，这（在某种意义上）可优化为一个 $\text{poly}(n)$ 因子；参见练习 30。

略证：设 $\varepsilon(n) \stackrel{\text{def}}{=} \varepsilon_G(n)$ 和 $l \stackrel{\text{def}}{=} \log_2(1/\varepsilon(n))$。回想 $\mathsf{E}[s(X_n)] = 0.5+\varepsilon(n)$，其中 $s(x) \stackrel{\text{def}}{=} \Pr[G(f(x), R_n) = b(x, R_n)]$（结论 2.5.2.1）。我们首先用一个更精确的分析代替结论 2.5.2.1。

结论 2.5.4.1：存在 $i\in\{1,\cdots,l\}$ 和基至少为 $(2^{i-1}\cdot\varepsilon(n))\cdot 2^n$ 的集合 $S_n \subseteq \{0,1\}^n$，使得对于所有 $x\in S_n$，有下式成立：

$$s(x) = \Pr[G(f(x), R_n) = b(x, R_n)] \geq \frac{1}{2} + \frac{1}{2^{i+1}\cdot l}$$

证明：设 $A_i \stackrel{\text{def}}{=} \{x: s(x) \geq \frac{1}{2}+\frac{1}{2^{i+1}l}\}$。对于任意非空集合 $S\subseteq\{0,1\}^n$，我们设 $\alpha(S) \stackrel{\text{def}}{=} \max_{x\in S}\{s(x)-0.5\}$ 和 $\alpha(\phi) \stackrel{\text{def}}{=} 0$。用反证法，设结论不成立（即对于 $i=1,\cdots,l$，有 $|A^i| < (2^{i-1}\cdot\varepsilon(n))\cdot 2^n$），我们可得：

$$\begin{aligned}
\mathsf{E}[s(X_n)-0.5] \leq\ & \Pr[X_n\in A_1]\cdot\alpha(A_1) \\
& + \sum_{i=2}^{l}\Pr[X_n\in(A_i\backslash A_{i-1})]\cdot\alpha(A_i\backslash A_{i-1}) \\
& + \Pr[X_n\in(\{0,1\}^n\backslash A_l)]\cdot\alpha(\{0,1\}^n\backslash A_l) \\
& < \varepsilon(n)\cdot\frac{1}{2} + \sum_{i=2}^{l}(2^{i-1}\cdot\varepsilon(n))\cdot\frac{1}{2^i l} + 1\cdot\frac{1}{2^{l+1}l} \\
& = \frac{\varepsilon(n)}{2} + (l-1)\cdot\frac{\varepsilon(n)}{2l} + \frac{2^{-l}}{2l} = \varepsilon(n)
\end{aligned}$$

这与 $\mathsf{E}[s(X_n)-0.5] = \varepsilon(n)$ 相矛盾。

给定任意一个 i 满足结论 2.5.4.1，设 $\varepsilon \stackrel{\text{def}}{=} 2^{-i-1}/l$，考虑对应的集合 $S_n \stackrel{\text{def}}{=} \{x: s(x)\geq 0.5+\varepsilon\}$。通过选择合适的参数，可得：对于所有 $x\in S_n$，算法 A' 的运行时间为 $O(n^3/\varepsilon^4)\cdot t_G(n)$，并且至少能以 $\frac{1}{2}$ 的概率从 $f(x)$ 求出 x。我们的下一个目标是提供一个更加有效的算法 A' 的实施方案，特别地，其运行时间为 $O(n^2/\varepsilon^2)\cdot(t_G(n)+\log(n/\varepsilon))$。

给经过修正后的算法 A' 输入 $y=f(x)$、参数 ε 和集合 $l=\log((n/\varepsilon^2)+1)$。在实际的描述中（见后面），采用实运算比采用布尔运算更方便。因此我们记 $b'(x,r) = (-1)^{b(x,r)}$ 和 $G'(y,$

$r)=(-1)^{G(y,r)}$。下面几个事实的证明作为练习来做。

事实 1：对于所有 x，有 $\mathsf{E}[b'(x,U_n)\cdot G'(f(x),U_n+e^i)]=s'(x)\cdot(-1)^{x_i}$，其中 $s'(x)\stackrel{\mathrm{def}}{=}2\cdot(s(x)-\frac{1}{2})$。（注：对于所有 $x\in S_n$，我们有 $s'(x)\geq 2\varepsilon$。）

事实 2：设 R 是一个均匀选择的 $l\times n$ 布尔矩阵，则对每一个 $v\neq u\in\{0,1\}^l\setminus\{0\}^l$，都有 vR 和 uR 是两两独立的并服从 $\{0,1\}^n$ 上的均匀分布。

事实 3：对每一个 $x\in\{0,1\}^n$，$v\in\{0,1\}^l$，都有 $b'(x,vR)=b'(xR^T,v)$。

利用这些事实可以得到如下结论。

结论 2.5.4.2：对任意 $x\in S_n$ 和均匀选择的 $l\times n$ 布尔矩阵 R，存在 $\sigma\in\{0,1\}^l$，使得对每一个 $1\leq i\leq n$，$\sum_{v\in\{0,1\}^l}b'(\sigma,v)\cdot G'(f(x),vR+e^i)$ 的符号至少以 $\frac{1}{2}$ 的概率等于 $(-1)^{x_i}$。

证明：令 $\sigma=xR^T$，与前面的事实相结合，对每一个 $v\in\{0,1\}^l\setminus\{0\}^l$，有 $\mathsf{E}[b'(xR^T,v)\cdot G'(f(x),vR+e^i)]=s'(x)\cdot(-1)^{x_i}$。这样，对每一个这样的 v，都有 $\Pr[b'(xR^T,v)\cdot G'(f(x),vR+e^i=(-1)^{x_i}]=\frac{1+s'(x)}{2}=s(x)$，利用事实 2，$l=\log((2n/\varepsilon^2)+1)$ 和 Chebyshev 不等式，可得结论。

注释的最后一部分：设 B 是一个 $2^l\times 2^l$ 的矩阵，(σ,v) 位置的元素为 $b'(\sigma v)$，设 $\overline{g}^i$ 为一个 2^l 维向量，第 v 位的值为 $G'(f(x),vR+e^i)$。这样向量 $B\,\overline{g}^i$ 的第 σ 个分量的值就等于 $\sum_{v\in\{0,1\}^l}b'(\sigma,v)\cdot G'(f(x),vR+e^i)$。

算法 A′的有效实现　当输入 $y=f(x)$ 和参数 ε 时，求逆算法 A' 使得 $l=\log((n/\varepsilon^2)+1)$，并如下进行：

1. 对 $i=1,\ldots,n$，计算 2^l 维向量 $\overline{g}^i$（如前定义）。
2. 对 $i=1,\ldots,n$，计算 $\overline{z}_i\leftarrow B\,\overline{g}^i$。

令 Z 是一个 $2^l\times n$ 实矩阵，第 i 个列等于 $\overline{z}_i$。

令 Z' 是一个 $2^l\times n$ 布尔符号矩阵，表示 Z 中元素的符号，特别地，Z' 的第 (i,j) 位等于 1 当且仅当 Z 的 (i,j) 位值是负的。

3. 观察 Z' 的全部行，输出满足 $f(z)=y$ 的第一行 z。

依据结论 2.5.4.2，前面算法由 $y=f(x)$ 求 x 的成功概率至少为 $\frac{1}{2}$。算法运行时间由步骤 1 和 2 决定，它们的运行时间分别为 $n\cdot 2^l\cdot O(t_G(n))=O((n/\varepsilon)^2\cdot t_G(n))$ 和

$$n\cdot O(l\cdot 2^l)=O((n/\varepsilon)^2\cdot\log(n/\varepsilon))$$

[①]最后，我们定义算法 A''。当输入 $y=f(x)$ 时，算法以 2^{-2j+1} 大的概率选择 $j\in\{1,\ldots,l\}$（在没有其他输出时就停止运行）。当输入 y 和参数 $\varepsilon=2^{-j-1}/l$ 时调用前面算法 A' 的实现方案，并如实返回 A' 的行为。A'' 的预期运行时间为

$$\sum_{j=1}^{l}2^{-2j+1}\cdot O\left(\frac{n^2}{(2^{-j-1}/l)^2}\right)\cdot(t_G(n)+\log(n\cdot 2^{j+1}l))=O(n^2\cdot l^2)\cdot t_G(n)$$

（假设 $t_G(n)=\Omega(l\log n)$）。令 $i\leq l$ 是满足结论 2.5.4.1 的指标（令 S_n 是相应的集合），我们考

---

① 利用矩阵 B 的特殊结构，可得给定一个向量 $\overline{w}$，结果 $B\overline{W}$ 能在时间 $O(l\cdot 2^l)$ 内得到计算。提示：B（*Sylvester* 矩阵）能被递归地写为 $S_k=\begin{pmatrix}S_{k-1} & S_{k-1}\\ S_{k-1} & \overline{S_{k-1}}\end{pmatrix}$。这里 $S_0=+1$，$\overline{M}$ 表示将 M 的 $+1$ 转变为 -1，反之亦然，因此 $\begin{pmatrix}S_{k-1} & S_{k-1}\\ S_{k-1} & \overline{S_{k-1}}\end{pmatrix}\begin{bmatrix}w'\\ w''\end{bmatrix}=\begin{bmatrix}S_{k-1}w'+S_{k-1}w''\\ S_{k-1}w'-S_{k-1}w''\end{bmatrix}$。这样，令 $T(k)$ 表示用一个 2^k 维向量乘 S_k 所需的时间，我们有 $T(k)=2\cdot T(k-1)+O(2^k)$，从而可得 $T(k)=O(k2^k)$。

虑 $j \geq i$(j 由算法 A''选择)的情况。依据结论 2.5.4.2,在这种情况下,对所有的 $x \in S_n$,当输入为 $f(x)$时算法 A'对 f 求逆的成功概率至少为$\frac{1}{2}$。利用 $i < l(=\log_2(1/\varepsilon(n)))$,可得

$$\Pr[A''(f(U_n)) = U_n] \geq \Pr[U_n \in S_n] \cdot \Pr[j \geq i] \cdot \frac{1}{2}$$

$$\geq 2^{i-1}\varepsilon(n) \cdot 2^{-2i+1} \cdot \frac{1}{2}$$

$$\geq \varepsilon(n) \cdot 2^{-l} \cdot \frac{1}{2} = \frac{\varepsilon(n)^2}{2}$$

从而命题得证。

注释 应用其他的技巧①,可以节约运行时间的一个 $\Theta(n)$因子,得到预期的运行时间 $O(n \cdot \log^3(1/\varepsilon_G(n))) \cdot t_G(n)$。

2.5.3* 核心函数

我们已经看到,所有的单向函数都能很容易地被修改以便有一个核心断言。换句话说,理论成果证实了关于原像的 1 比特的信息很难从函数的值得到近似。更强的结论可能会认为,几比特的关于原像的信息是更难近似的。例如,以不可忽略的大于$\frac{1}{4}$的概率来猜测这个数对是不可能的,在这种意义下,我们说一些特定位对是很难被近似的。实际上,一般来说,我们采用稍微不同的方法并且要求这些位的真实值与随机值很难被区分。也就是说,如果没有一个有效的算法能够区分$(f(x),h(x))$和$(f(x),r)$,这里 r 是一个长度为$|h(x)|$的随机串,我们就称多项式时间函数 h 是函数 f 的核心。关于有效分辨力概念的进一步讨论可以参考 3.2 节。我们做如下的简化假设,即 h 是长度正则的(见下节)。

定义 2.5.5(核心函数):设 $h:\{0,1\}^* \to \{0,1\}^*$ 是一个多项式时间可计算函数,并对所有$|x|=|y|$满足$|h(x)|=|h(y)|$,又设 $l(n) \stackrel{\text{def}}{=} |h(1^n)|$。如果对于每一个概率多项式时间算法 D',每一个正多项式 $p(\cdot)$和所有足够大的 n,都有

$$|\Pr[D'(f(X_n),h(X_n)) = 1] - \Pr[D'(f(X_n),R_{l(n)}) = 1]| < \frac{1}{p(n)}$$

其中,X_n和 $R_{l(n)}$ 是两个独立的随机变量,X_n服从$\{0,1\}^n$上的均匀分布,$R_{l(n)}$服从$\{0,1\}^{l(n)}$上的均匀分布,则称函数 h 为函数 f 的核心。

当 $l \equiv 1$ 时,定义 2.5.5 和定义 2.5.1 是等价的;具体讨论见后面的引理 2.5.8 和习题 31。

具有对数长度的简单核心函数(即 $l(n)=O(\log n)$)已知的有 RSA, Rabin 和 DLP 类,如果相应的这些类是单向的话。例如,只要 RSA 类是单向的,输出的对数多数低有效位的函数就是 RSA 类的核心函数。也就是说,若 RSA 类是单向的,则给定 $\mathrm{RSA}_{N,e}(x)=x^e \bmod N$,要想区分 $O(\log|N|)$的 x 的低有效位和均匀分布的 $O(\log|N|)$位长的串是不可行的(对 Rabin 类

---

① 令 $2^l=O(1/\varepsilon^2)$(而不是 $2^l=O(n/\varepsilon^2)$),进一步修正算法 A'。在新的设定下,我们可以以固定的概率正确地求 x 的一些固定位(而不是所有位)。如果 x 是一个渐进纠错码的码字(*cf.* [138]),这一点就可能满足。为了避免这个假设,我们修正算法 A'使得它能够求出 x 位的异或位(而不是 x 的个别位)。特别地,我们使用一个渐进的好的线性码(即以恒定的比率纠正固定部分的错误,并且有一个有效的解码算法)。这样,修正算法 A'在这种码下在对 x 的编码过程中可以正确地求出位的固定部分,并应用解码算法就能够恢复 x。

和 DLP 类，有类似的结论)。下面是一个更一般的结论。

定理 2.5.6：设 f 是任意的强单向函数，g_2 如下定义：$g_2(x,s) \stackrel{\text{def}}{=} (f(x),s)$，这里 $|s|=2|x|$。[①] 令 $b_i(x,s)$ 表示二进制向量 x 和 $(s_{i+1},\cdots,s_{i+n})$ 的模 2 内积，这里 $s=(s_1,\cdots,s_{2n})$。那么对任意常数 $c>0$，函数 $h(x,s) \stackrel{\text{def}}{=} b_1(x,s)\cdots b_{l(|x|)}(x,s)$ 是函数 g_2 的核心函数，这里 $l(n) \stackrel{\text{def}}{=} \min\{n,\lceil c\log_2 n\rceil\}$。

定理证明时结合利用明确函数 h 的结构的命题和有关核心函数的一般引理。简单地说，命题将给定 $g(x,r)$ 时的近似问题转变为给定 $g_2(x,s)$ 时 $h(x,s)$ 非空子集的位的异或的近似问题，这里 b 和 g 都是前面提到的核心和单向函数。既然知道断言 $b(x,r)$ 不能由 $g(x,r)$ 来近似，就可推出没有一个 $h(x,s)$ 的位的异或可由 $g_2(x,s)$ 来近似。一般的引理告诉我们，对每一个“对数缩减”函数 h'(即 h' 满足 $|h'(x)|O(\log|x|)$)，函数 h' 是函数 f' 的核心，当且仅当 h' 位的任意非空子集的异或不能够由 f' 的值来近似，下面是正式的陈述和两个结论的证明。

命题 2.5.7：设 f,g_2,l 和 b_i 与定理 2.5.6 所述的一样。设 $\{I_n\subseteq\{1,2,...,l(n)\}\}_{n\in\mathbb{N}}$ 是任意的非空集序列，并设 $b_{I_{|x|}}(x,s) \stackrel{\text{def}}{=} \bigoplus_{i\in I_{|x|}} b_i(x,s)$。那么对于每一个概率多项式算法 A'、每一个正多项式 $p(\cdot)$ 和所有足够大的 n，有

$$\Pr[A'(I_n,g_2(U_{3n}))=b_{I_n}(U_{3n})]<\frac{1}{2}+\frac{1}{p(n)}$$

其中 U_{3n} 是一个在 $\{0,1\}^{3n}$ 上服从均匀分布的随机变量。

证明：通过可约性理论可以获证。设 X_n,R_n 和 S_{2n} 是相互独立的分别在 $\{0,1\}^n$，$\{0,1\}^n$ 和 $\{0,1\}^{2n}$ 上服从均匀分布的随机变量。我们指出给定 $f(X_n,R_n)$ 来近似 $b(X_n,R_n)$ 的问题可转化为给定 $(f(X_n),S_{2n})$ 来近似 $b_{I_n}(X_n,S_{2n})$ 的问题。通过观察，对每一个 $|s|=2\cdot|x|$ 和每一个 $I\subseteq\{1,...,l(n)\}$，都有

$$b_I(x,s)=\bigoplus_{i\in I}b_i(x,s)=b(x,\bigoplus_{i\in I}\mathrm{sub}_i(s))$$

这里 $\mathrm{sub}_i(s_1,...,s_{2n}) \stackrel{\text{def}}{=} (s_{i+1},...,s_{i+n})$。读者可进一步证明，对于所有非空的 $I\in\{1,...,l(n)\}$，随机变量 $\bigoplus_{i\in I}\mathrm{sub}_i(S_{2n})$ 服从 $\{0,1\}^n$ 上的均匀分布，并且给定一个串 $r\in\{0,1\}^n$ 和一个集合 I，可以在集合 $\{s:\bigoplus_{i\in I}\mathrm{sub}_i(s)=r\}$ 上有效地选择一个均匀分布的串。这两个结论的证明留做练习。[②]

现在做一个相反的假设，假设存在一个有效算法 A'、一个多项式 $p(\cdot)$，以及一个无限集合(即 I_n)序列和对应的 n 集合，使得

$$\Pr[A'(I_n,g_2(U_{3n}))=b_{I_n}(U_{3n})]\geq\frac{1}{2}+\frac{1}{p(n)}$$

我们首先看到对于满足前述不等式的 n，我们能很容易地找到一个集合 I，满足

---

① 事实上，我们可以应用 $|s|=|x|+l(|x|)-1$，这里 $l(n)=O(\log n)$。在目前的描述中，s_1 和 $s_{n+l(n)+1},\cdots,s_{2n}$ 都没有用到。利用现有的公式，可以在定义 g_2 时不必再详细说明 l。

② 给出任意非空集合 I 和 $r=r_1\cdots r_n\in\{0,1\}^n$，考虑如下过程，其中 k 是 I 的最大元素。首先，均匀选择 $s_1,...,s_k,s_{k+n+1},...,s_{2n}\in\{0,1\}$。其次，从 $i=1$ 到 $i=n$，确定 s_{k+i} 以便 $\bigoplus_{j\in I}s_{i+j}=r_i$(即：$s_{k+i}\leftarrow r_i\bigoplus(\bigoplus_{j\in I\setminus\{k\}}s_{j+i})$，其中，由于 $j<k$，相关的 s_{i+j} 都已经确定。上述过程确定了串 $s_1,\cdots,s_{2n}$ 它们在 2^n 中均匀分布，且满足 $\bigoplus_{i\in I}\mathrm{sub}_i(s)=r$，这是因为选取 r 有 2^n 种可能性，

$$PI \stackrel{\text{def}}{=} \Pr[A'(I, g_2(U_{3n})) = b_I(U_{3n})] \geq \frac{1}{2} + \frac{1}{2p(n)}$$

特别地，我们可以试验所有可能的 I 并估计每一个 I 的 P_I 值（通过随机试验），选择使估计值最大的 I。（注：进行 poly(n)次实验，我们可以以至多 $1/4p(n)$ 的偏差和 2^{-n} 的错误概率估计每一个可能的 $2^{l(n)}-1=\text{poly}(n)$ 个不同的 P_I）。

我们现在提出一个由 $y \stackrel{\text{def}}{=} f(x)$ 和 r 来近似 $b(x,r)$ 的算法。当输入 y 和 r 时，算法首先找到一个如前所述的集合 I，（这个阶段仅仅是依赖 $n \stackrel{\text{def}}{=} |x| = |r|$）。一旦找到了 I，算法就可以均匀地选择串 s，使得 $\bigoplus_{i\in I}\text{sub}_i(s) = r$ 并输出 $A'(I,(y,s))$。

注意到对均匀分布的 $r \in \{0,1\}^n$，算法选择的串 s 服从 $\{0,1\}^{2n}$ 上的均匀分布。$b(x,r) = b_I(x,s)$ 对这个算法的成功概率的评估留做练习。■

下面的引理提供了一类由区分 $f(X_n, h(X_n))$ 和 $(f(X_n), R_{l(n)})$ 的算法到另一算法的转变方法，即给定 $f(X_n)$ 和 $\{1,\ldots,l(n)\}$ 的一个随机非空子集 I，判断 I 中 X_n 的位的异或的算法。

引理 2.5.8（计算异或引理）：设 f 和 h 是任意的长度正则的函数，并设 $l(n) \stackrel{\text{def}}{=} |h(1^n)|$。令 D 是任意算法，并记 $p \stackrel{\text{def}}{=} \Pr[D(f(X_n), h(X_n)) = 1]$ 和 $q \stackrel{\text{def}}{=} \Pr[D(f(X_n), R_{l(n)}) = 1]$，这里 X_n 和 $R_{l(n)}$ 是相互独立的并分别服从 $\{0,1\}^n$ 和 $\{0,1\}^{l(n)}$ 上的均匀分布。我们考虑一个特殊的算法，记 $G \stackrel{\text{def}}{=} G_D$，这里 D 是一个子程序。特别地，当输入 y 和 $S \subseteq \{1,\ldots,l(n)\}$ 时，算法 G 在 $\{0,1\}^{l(n)}$ 上均匀地选择 $r = r_1\ldots,r_{l(n)}$，并输出 $D(y,r)\oplus 1 \oplus (\bigoplus_{i\in S} r_i)$，则：

$$\Pr[G(f(X_n), I_l, l(n)) = \bigoplus_{i\in I_l}(h_i(X_n))] = \frac{1}{2} + \frac{p-q}{2^{l(n)}-1}$$

这里 I_l 是随机选择的 $\{1,\ldots,l(n)\}$ 的非空子集，$h_i(x)$ 表示 $h(x)$ 的第 i 位。

从而对于对数缩减的函数 h，能够区分随机变量 $(f(X_n), h(X_n))$ 和 $(f(X_n), R_{l(n)})$ 的有效算法的存在性就意味着存在一个有效的算法，该算法能以不可忽略的概率由 $f(X_n)$ 近似 $h(X_n)$ 的位的随机非空子集的异或。另一方面，很明显任何能够由 f 的值近似 h 的位的随机非空子集的异或的算法都能被很容易地修正，以区分 $(f(X_n), h(X_n))$ 和 $(f(X_n), R_{l(n)})$。因此，对数收缩的函数 h 是函数 f 的核心，当且仅当 h 的位的任意非空子集的异或不能由 f 的值来近似。

证明：需要做的全部事情就是估计算法 G（作为 $p-q$ 的一个函数）的成功概率。首先固定 $x \in \{0,1\}^n$ 并估计 $\Pr[G(f(x), I_l, l) + \bigoplus_{i\in I_l}(h_i(x))]$，这里 I_l 是均匀选择的 $\{1,\ldots,l\}$ 的非空子集且 $l \stackrel{\text{def}}{=} l(n)$。剩下的步骤是对所有的 x 做一个简单的平均。

令 C 表示所有 $\{1,\ldots,l\}$ 的非空子集的集合。对每一个 $S \in C$，定义一个关系 $\equiv_S$ 使得 $y \equiv_S z$，当且仅当 $\bigoplus_{i\in S} y_i = \bigoplus_{i\in S} z_i$，这里 $y = y_1,\cdots y_l$，$z = z_1,\cdots z_l$。注意到对每一个 $S \in C$ 和 $z \in \{0,1\}^l$，关系 $y \equiv_S z$ 对 y 的 2^{l-1} 个位都成立。回想 G 的定义，当输入 $(f(x), S, l)$ 和随机选择 $r = r_1\cdots r_l \in \{0,1\}^l$ 时，算法 G 输出 $D(f(x),r)\oplus 1 \oplus (\bigoplus_{i\in S} r_i)$。而后者等于 $\bigoplus_{i\in S}(h_i(x))$，当且仅当以下两个不相交的事件中有一个发生：

事件 1：$D(f(x),r) = 1$ 和 $r \equiv_S h(x)$。

事件 2：$D(f(x),r) = 0$ 和 $r \not\equiv_S h(x)$。

通过前述的讨论和基本处理，我们得到：

$$s(x) \stackrel{\text{def}}{=} \Pr[G(f(x), I_l, l) = \bigoplus_{i\in I_l}(h_i(x))]$$

$$= \frac{1}{|C|} \cdot \sum_{S \in C} \Pr[G(f(x), S, l) = \bigoplus_{i \in S}(h_i(x))]$$

$$= \frac{1}{|C|} \cdot \sum_{S \in C} (\Pr[\text{event } 1] + \Pr[\text{event } 2])$$

$$= \frac{1}{2 \cdot |C|} \cdot \sum_{S \in C} (\Pr[\Delta(R_l) = 1 \mid R_l \equiv_S h(x)] + \Pr[\Delta(R_l) = 0 \mid R_l \not\equiv_S h(x)])$$

这里 R_l 服从 $\{0,1\}^l$ 上的均匀分布(代表算法 G 的随机选择),$\Delta(r)$ 是随机变量 $D(f(x), r)$ 的简写。剩余的分析比较直观,但很冗长,此处省略并无大碍。

$$s(x) = \frac{1}{2} + \frac{1}{2|C|} \cdot \sum_{S \in C} (\Pr[\Delta(R_l) = 1 \mid R_l \equiv_S h(x)] - \Pr[\Delta(R_l)$$

$$= 1 \mid R_l \not\equiv_S h(x)])$$

$$= \frac{1}{2} + \frac{1}{2|C|} \cdot \frac{1}{2^{l-1}} \cdot \left(\sum_{S \in C} \sum_{r \equiv_S h(x)} \Pr[\Delta(r) = 1] - \sum_{S \in C} \sum_{r \not\equiv_S h(x)} \Pr[\Delta(r) = 1]\right)$$

$$= \frac{1}{2} + \frac{1}{2^l \cdot |C|} \cdot \left(\sum_r \sum_{S \in EQ(r, h(x))} \Pr[\Delta(r) = 1] - \sum_r \sum_{S \in NE(r, h(x))} \Pr[\Delta(r) = 1]\right)$$

其中,$EQ(r,z) \stackrel{\text{def}}{=} \{S \in C : r \equiv_S z\}$,$NE(r,z) \stackrel{\text{def}}{=} \{S \in C : r \not\equiv_S z\}$。我们可以看到,对于所有 $r \neq z$,有 $|NE(r,z)| = 2^{l-1}$(和 $|EQ(r,z)| = 2^{l-1} - 1$)成立。另一方面,对于所有 z,有 $EQ(z,z) = C$(和 $NE(z,z) = \emptyset$)成立。因此得出:

$$s(x) = \frac{1}{2} + \frac{1}{2^l |C|} \sum_{r \neq h(x)} ((2^{l-1} - 1) \cdot \Pr[\Delta(r) = 1] - 2^{l-1} \cdot \Pr[\Delta(r) = 1])$$

$$+ \frac{1}{2^l |C|} \cdot |C| \cdot \Pr[\Delta(h(x)) = 1]$$

$$= \frac{1}{2} - \frac{1}{2^l |C|} \sum_{r \neq h(x)} \Pr[\Delta(r) = 1] + \left(\frac{1}{|C|} - \frac{1}{2^l |C|}\right) \cdot \Pr[\Delta(h(x)) = 1]$$

这里最后的等式采用 $|C| = 2^l - 1$(即 $\frac{1}{2^l} = \frac{1}{|C|} - \frac{1}{2^l |C|}$)重新组织各项并替换 Δ,可得:

$$s(x) = \frac{1}{2} + \frac{1}{|C|} \cdot \Pr[\Delta(h(x)) = 1] - \frac{1}{2^l |C|} \sum_r \Pr[\Delta(r) = 1]$$

$$= \frac{1}{2} + \frac{1}{|C|} \cdot (\Pr[D(f(x), h(x)) = 1] - \Pr[D(f(x), R_l = 1)])$$

最后,对 x 取期望值,得到:

$$E[s(X_n)] = \frac{1}{2} + \frac{1}{|C|} \cdot (\Pr[D(f(X_n), h(X_n)) = 1] - \Pr[D(f(x_n), R_l = 1])$$

$$= \frac{1}{2} + \frac{1}{2^l - 1} \cdot (p - q)$$

引理得证。■

2.6* 单向函数的有效放大

在定理 2.3.2 中阐述的将弱单向函数放大为强单向函数没有实际价值。让我们回顾一下,放大变换是将一个对长为 n 的字符串的显著(即 $\frac{1}{p(n)}$)部分难求逆的函数 f,转变为一种对

所有长度为$n^2p(n)$的字符串除去可忽略的部分难求逆的变换 g。特别地，能将一种对长为$n^2p(n)$的字符串的$\varepsilon(n)$部分在运行时间$T(n)$内对g求逆的算法，转变成一种对长为n的字符串的$1-\frac{1}{p(n)}$部分，在运行时间$poly(p(n),n,\frac{1}{\varepsilon(n)\cdot T(n)}$内对$f$求逆的算法。因此，如果在实际中$f$对长为1000的字符串的1%难求逆。那么我们可以说，在实际中g对几乎所有长度为1000000000的字符串难求逆。与此构成对比的是，正如稍后将要给出的那样，单向函数的一种有效放大，应当是将(弱单向)变换f在长度为n的字符串上的求逆困难性，关联到(强单向)变换g在长度为$O(n)$的字符串上的求逆困难性，而不是变换g在长度为$poly(n)$的字符串上的求逆困难性。因此，我们可以这样断言：如果在实际中当输入长度1000的字符串时，对其中的1%，f很难求逆，那么对几乎所有的长度为5000的字符串g难求逆。下面给出的更加自然的定义是为了对单向函数的放大进行更加一般化的讨论。

定义 2.6.1(量化单向性)：令$T:\mathbb{N}\to\mathbb{N}$，$\varepsilon:\mathbb{N}\to\mathbb{R}$为多项式时间可计算的变换。如果对任何一种算法$A'$，对于运行时间限制于$T(\cdot)$和足够大的$n$，有

$$\Pr[A'(f(U_n))\notin f^{-1}(f(U_n))]>\varepsilon(n)$$

则称多项式时间可计算的变换$f:\{0,1\}^*\to\{0,1\}^*$为关于时间$T(\cdot)$的$\varepsilon(\cdot)$-单向。

使用这一术语，我们回顾一下我们已经了解的单向函数的放大。如果存在多项式$p(\cdot)$，使得f是关于多项式时间的$\frac{1}{p(\cdot)}$-单向，则变换f是弱单向性的。[①] 如果存在多项式$q(\cdot)$，使得f是关于多项式时间的$(1-\frac{1}{q(\cdot)})$-单向，则变换f是强单向性的。(恒等变换只对于线性时间是0-单向，而对于线性时间没有任何变换是$(1-\exp(\cdot))$-单向的。[②])定理2.3.2的放大结果可被一般性地重新表述如下：如果存在多项式p和(多项式时间可计算的)变换f，且f是关于时间$T(\cdot)$的$\frac{1}{p(\cdot)}$-单向，那么就存在(多项式时间可计算的)变换g，且g是关于时间$T'(\cdot)$的强单向，其中$T'(n^2p(n))=T(n)$，换言之，对某些$\varepsilon>0$且满足$(n^2\cdot p(n))^{\varepsilon}\le n$，有$T'(n)=T(n^{\varepsilon})$。与此对照的是，正如稍后给出的那样，单向变换的有效放大应表述为在$T'(O(n))=T(n)$(换言之，对某些$\varepsilon>0$，有$T'(n)=T(\varepsilon\cdot n)$)的条件下上述成立。对正则单向变换可以得到这样的结果。如果存在多项式时间可计算的变换$m:\mathbb{N}\to\mathbb{N}$和多项式$p(\cdot)$，使得在变换$f$值域范围内的任意$y$，变换$f$下的$y$的原像数目(长度为$n$)介于$\frac{m(n)}{p(n)}$和$m(n)\cdot p(n)$之间，则称变换$f$是正则的。本书中，我们只研究单向置换的结果(即等长度的1-1变换)。

定理 2.6.2(单向置换的有效放大)：令$p(\cdot)$为一多项式，$T:\mathbb{N}\to\mathbb{N}$为一变换，假设多项式时间可计算的置换$f$是关于时间$T(\cdot)$的$\frac{1}{p(\cdot)}$-单向，那么存在常数$\gamma>1$、多项式$q$与多项式时间可计算的置换$F$，使得对于任意定义于$N\to[0,1]$上的多项式时间可计算的变换，置换$F$是关于时间$T'_{\varepsilon}(\cdot)$的$(1-\varepsilon(\cdot))$-单向，其中$T'_{\varepsilon}(\gamma\cdot n)\stackrel{\text{def}}{=}\frac{\varepsilon(n)^2}{q(n)}\cdot T(n)$。

常数γ只取决于多项式p($\cdot$)。

---

① 此处以及今后，关于多项式时间意指对于任意的多项式T，关于时间T。

② 恒等变换在线性时间内可以零错误概率求逆。另一方面，对于任意变换f，给定y，输出$0^{|y|}$的算法对f求逆$f(U_n)$的错误概率至多为$1-2^{-n}<1-\exp(-n)$。

2.6.1　构造

单向置换 f 的放大的关键在于在各种不同的论题下使用 f。在定理 2.3.2 的证明中，f 使用了不相关的参数（输入的不连续的各部分）。这使得证明相对简单，同时也使得其构造非常低效。相对地，在此定理的证明里表达的构造中，我们为单向函数 f 指定了相关的参数。首先想到的是迭代 f 多次，每次使用前一次运算的结果。然而如果求逆算法的简单实例连续地被 f 映射到其自身，那么这种方法将无效。我们不能仅仅期望这种情况不发生。因此接下来想到的就是使 f 的逐次结果随机化。只使用少量的随机化操作是很重要的，因为随机化操作会被编码至我们构造的变换的参数里。f 逐次结果之间的随机化操作是以在扩张图上的随机行走的形式进行的。因此我们有必要解释一下关于这些图和在图上的随机行走的概念。

称图 $G=(V, E)$ 是一个 (n, d, c)-扩张图，如果图中有 n 个顶点（即 $|V|=n$），V 中每个顶点时度数为 d（即 G 是 d-正则的），且 G 具备以下的扩张属性（扩张因子 $c>0$）：对任一子集 $S\subset V$，如 $|S|\leq\frac{n}{2}$，则 $|N(S)|\geq(1+c)\cdot|S|$，式中 $N(S)$ 表示 S 中邻接顶点的集合，即 $N(S)\stackrel{\text{def}}{=}\{u\in V: \exists v\in S \text{ s.t. } (u,v)\in E\}$。[①] 显式构造 (d, c)-扩张图的含义是图 $\{G_n\}_{n\in\mathbb{N}}$ 的族，其中 G_n 是 $(2^n, d, c)$-扩张图，且存在若输入为扩张图中的一个顶点时，则输出为其邻接顶点的列表的多项式时间算法，而 G_n 中的顶点是以长度为 n 的二进制字符串表示的。我们强调指出不仅算法本身，而且常数 $d\in\mathbb{N}$，$c>0$ 的条件需对族中的所有图均成立。如此的扩张图族是存在的。在图上随机行走的含义是：随机地从顶点中选取一个作为出发点，每一步再随机选取当前顶点的邻接顶点，每次选取服从均匀概率分布，如此所经过的顶点序列称为随机行走。扩张属性隐含地表示（并非不证自明）在扩张图上随机行走的顶点具备强烈的"随机属性"。尤其要指出的是，对顶点集合中恒定密度的每一子集和任一 l，$O(l)$-步长的行走中没有顶点包含于该子集的概率至多是 2^{-l}（即相当于我们独立地选取 $O(l)$ 个顶点的情况），其中 O 标记法中的常数仅取决于图的扩张因子。

我们提醒读者注意，我们对逐次使用置换 f 很感兴趣，同时在逐次结果之间交叉插入随机的间隔。因此，在对前一次结果进行置换 f 的操作之前，我们在扩张图上进行一次随机的行走。也即，我们将给定的单向函数的概念与扩张图中的顶点联系到一起。我们的构造方法交替地使用到了给定的单向置换 f 和从一顶点随机移动至其邻接顶点中的一个。一个关键的结论是，对一个扩张图中顶点的任意置换的构造过程生成了另一个（具备相同扩张属性的）扩张图。结合在扩张图上随机行走的特性和"可约"参数，下面的构造可以用来以一种有效的方式放大给定置换的单向性。（我们强调指出，定理 2.6.2 不能被下述构造的直接结果所证明，参见 2.6.2 小节。）

构造 2.6.3：令 $\{G_n\}_{n\in\mathbb{N}}$ 为 d-正则图的族，则 G_n 中的顶点集合为 $\{0,1\}^n$，每个顶点均有自环。考虑对连接至每个顶点的边进行标记的事件（使用标记 $1, 2, \ldots, d$）。定义 $g_l(x)$ 为从顶点 x 出发沿标记为 l 的边可到达的顶点。令 $f:\{0,1\}^*\to\{0,1\}^*$ 为一等长度的 1-1 变换，令

---

① 我们使用了某种意义上非标准的定义。以因子 $c>0$ 扩张的标准定义为，对任一满足 $S\subset V$ 且 $|S|\leq\frac{n}{2}$ 的 S，成立 $|N'(S)|\geq c\cdot|S|$，$N'(S)$ 表示在 $V\setminus S$ 中并在 S 中有邻接顶点的顶点（即 $N'(S)\stackrel{\text{def}}{=}\{u\in V\setminus S: \exists v\in S \text{ s.t. } (u,v)\in E\}$）。在标准定义下，每个 (n,d,c)-扩张图都可被转换为 $(n,d+1,c)$-扩张图（例如，通过增加自环）。

λ 为($\{1,2,\ldots,d\}$上的)空序列。则对于任一 $k\geq 0, x\in\{0,1\}^n, \sigma_1,\sigma_2,\ldots,\sigma_k\in\{1,2,\ldots,d\}$,定义 $F(x,\lambda)=x$,并且

$F(x,\sigma_1\sigma_2\ldots\sigma_k)=\sigma_1,F(g_{\sigma_1}(f(x)),\sigma_2\ldots,\sigma_k)$,即

$$F(x,\sigma_1\sigma_2\ldots\sigma_k)=\sigma_1\sigma_2,\ldots,\sigma_k,y$$

式中 $y=g_{\sigma k}(f(\ldots(g_{\sigma 2}(f(g_{\sigma 1}(f(x)))))\ldots))$

对任一 $k:\mathbb{N}\to\mathbb{N}$,定义 $F_k(\alpha)\stackrel{\text{def}}{=}F(x,\sigma_t,\ldots,\sigma_t)$,$\alpha$ 被解析为$(x,\sigma_1,\ldots,\sigma_t)$,导出 $t=k(|x|),\sigma_i\in\{1,2,\ldots,d\}$。

显然,F_k是 1-1 且等长度的。从 x 得到 y 的过程(对于 $k=4$)在图 2.3 中给出:划圈的 f 表示单向函数 f 的操作,标记 g 的方块表示在扩张图中行进一步(通过辅助输入 σ_i 指定方向)。

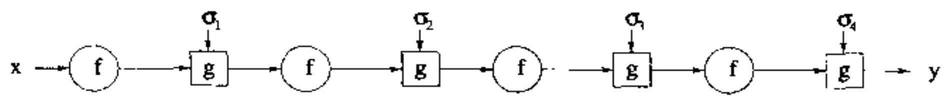


图 2.3 构造 2.6.3 的本质

2.6.2 分析

构造 2.6.3 的"困难度放大"性质在如下的命题中予以陈述。

命题 2.6.4: 令$\{G_n\}$、$f\{0,1\}^*\to\{0,1\}^*$,$k:\mathbb{N}\to\mathbb{N}$,$F_k$如构造 2.6.3 所述。令 $d\in N$,$c>0$,l 是常数,且 $\alpha:\mathbb{N}\to\mathbb{R}$ 和 $T:\mathbb{N}\to\mathbb{N}$ 是使得下述条件成立的变换:

1. 图$\{G_n\}_{n\in\mathbb{N}}$族是(d,c)-扩张图的显式构造族。

2. 置换 f 是关于时间 $T:\mathbb{N}\to\mathbb{N}$ 的 $\alpha'(\cdot)$-单向$(\alpha'(n)=\alpha(n)+2^{-n})$,且是多项式时间可计算的。

3. 变换 $\alpha:\mathbb{N}\to\mathbb{R}$ 是多项式时间可计算的。

4. $l\geq\frac{4+c^2}{c^2}\cdot d$。

由此,置换 F_k是多项式时间可计算的,且对于任一多项式时间可计算的 $\varepsilon:\mathbb{N}\to\mathbb{R}$,置换 F_k 是关于时间 $T':\mathbb{N}\to\mathbb{N}$ 的$((1-\varepsilon(\cdot))\beta(\cdot))$-单向。其中,

$$\beta(n+k(n)\cdot\log_2 d)\stackrel{\text{def}}{=}1-\left(1-\frac{\alpha(n)}{2}\right)^{k(n)/l}$$

$$T'(n+k(n)\cdot\log_2 d)\stackrel{\text{def}}{=}\frac{(\varepsilon(n)\cdot\alpha(n))^2}{O(n+k(n))^3}\cdot T(n)$$

对 $k(n)=3l\cdot n$ 和 $\alpha(n)=1/\text{poly}(n)$,我们得到 $\beta(O(n))=1-(1-0.5\cdot\alpha(n))^{3n}$ 和 $T'(O(n))=\text{poly}(\varepsilon(n)/n)\cdot T(n)$。尤其指出的是,对 $\alpha(n)=o(1/n)$有 $\beta(O(n))\approx 1.5n\cdot\alpha(n)$,对 $\alpha(n)\leq 1/2n$ 有 $\beta(O(n))>1.02n\cdot\alpha(n)$,对常数 α 有 $\beta(O(n))>1-2^{-\Omega(n)}$。

定理 2.6.2 的证明: 设 δ 是多项式 $p(\cdot)$的度(由假设 f 是$\frac{1}{p(\cdot)}$-单向所确定),重复应用命题 2.6.2 $\delta+1$ 次,则定理 2.6.2 成立。在命题的所有应用中,我们令 $k(n)\stackrel{\text{def}}{=}=3l\text{n}$,在最初的 δ 次应用中,我们令 $\varepsilon(n)=0.01$。对 $i\leq\delta$ 命题的第 i 次应用的变换结果是$\frac{1}{2n^{\delta-i}}$单向的。特别地,δ 次应用之后,变换结果是$\frac{1}{2}$-单向的(看起来$\frac{1}{2}$一单向的概念值得特别注意,也值得命名一个类似于"大多数单向"的名称)。在最后一次(即第 $\delta+1$ 次)应用中,我们令 $\varepsilon(n)=\in(n)$。

命题的最后一次(即第 $\delta+1$ 次)应用的变换结果满足定理 2.6.2 的陈述。■

对命题 2.6.4 的证明回顾:命题本身是通过结合两种不同类型的参数得到证明的,参数的主要部分在引理 2.6.5 和下面的引理 2.6.6 中予以陈述。引理 2.6.5 是关于在扩张图上随机行走行为的纯组合性引理。引理 2.6.6 预设这种行为(在图 $\{G_{f,n}\}$ 上随机行走,在下面定义)并有序地使用之以建立命题 2.6.4。引理 2.6.6 是通过一个对定理 2.3.2 进行一般化的可约参数进行证明的。我们以组合数学开始。

组合数学 首先注意到我们并非对在图 G_n 上的随机行走感兴趣,而是对在图 $G_{f,n}$ 上的随机行走感兴趣,其中 $G_{f,n}\stackrel{\text{def}}{=}(\{0,1\}^n,E_{f,n})$ 是通过定义 $G_n=(\{0,1\}^n,E_n)$ 和 $E_{f,n}\stackrel{\text{def}}{=}\{(u,v):(f(u),v)\in E_n\}$ 而得到的。首先得到的结论是 $G_{f,n}$ 保持了 G_n 的扩张性质,因为 f 是 $\{0,1\}^n$ 上的置换(一般地,对任何图 $G=(V,E)$,若 $f:V\to V$ 是 1-1 的,则类似的定义有 $G_f=(V,E_f)$,保持了 G 的扩张性质[①])。下一个组合的步骤显示了对假设中的 c 和 d,每个 G_n 的邻接矩阵中最大的两个特征值的比率(以绝对值形式)以 1 为上界。也即,对某个 $\rho<1$ 和所有的 n、G_n 的特征值比率至多为 ρ(这是通过正则图的扩张常数与其邻接矩阵的特征值比率之间的关系得到表示的,具体地,$\rho\leq1-\frac{c^2}{(4+c^2)\cdot d}$)。其次得到的结论是:令 P_l 表示 $G_{f,n}$ 中 l-边长的路径的集合,令 $G_{f,n}^l=(\{0,1\}^n,P_l)$,则图 $G_{f,n}$ 的特征值比率最大为 ρ^l。通过关于 l 的假设和界定 ρ,有 $\rho^l<\frac{1}{2}$。主要的组合数学步骤在下面的引理中陈述。[②]

引理 2.6.5(随机行走引理):令 G 是正则图,其邻接矩阵中第一个特征值与第二个特征值的绝对值比率小于 1/2。令 S 为图中顶点的量度 μ 的子集,则在 G 上长度为 t 的随机行走命中 S 的概率至少为 $1-(1-0.5\cdot\mu)^t$。

证明思想:因为证明与本书的主题没有太大的关系,所以在这里我们只是给出证明中涉及的粗略的思想。证明使用到通过将邻接矩阵 G 与 G 的度相除得到的随机矩阵,且将图的顶点集合的概率分布视作该矩阵的(正交)特征值的线性组合。新矩阵的特征值的比率与邻接矩阵 G 的相等。此外,最大的特征值是 1,与其关联的特征向量是均匀分布的。

沿着随机路径行走,从上述分析我们对与不经过集合 S 的随机路径相关的概率块定界。在每一步骤,第一个特征向量方向的当前分布的分量以因数 μ 减少其权重(减少的原因是当前步骤中进入 S 的路径的分数)。对第二个特征向量应用此界,可以看到每一步骤中其他要素的 L_2-范数以因数 2 减少(因此剩余分布被"推"向第一个特征向量的方向)。直观地,通过集合 S 的事件成为剩余分布的一种过滤筛,但是这个过滤筛仅当剩余分布接近于平均分布时有效,且在扩张图上的下一个随机步骤时仍保留。

下面给出紧密遵循上述直观分析的(粗略的)正式分析。不过,简单分析仅能给出比论题较弱些的界。这个弱些的界并不能满足我们的目标,因为它仅在 $\mu\geq\frac{1}{4}$ 时有意义(而我们也需要联系小得多的 μ 值,特别地,为 $1/\mu$,即图的大小的多对数(*poly-logarithmic*)。

---

① 换言之,令 $E_f\stackrel{\text{def}}{=}\{(u,v):(f(u),v)\in E\}$,并记 $N(S)\stackrel{\text{def}}{=}\{v\in V:\exists u\in S\text{ s.t. }(u,v)\in E\}$,$N_f(S)\stackrel{\text{def}}{=}\{v\in V:\exists u\in S\text{ s.t. }(u,v)\in E_f\}$,则 $N_f(S)=\{v\in V:\exists f(u)\in f(S)\text{ s.t. }(f(u),v)\in E\}=N(f(S))$,其中 $f(S)\stackrel{\text{def}}{=}\{f(u):u\in S\}$,利用 f 的 1-1 性质,得到 $|f(S)|=|S|$,因此断言成立(即如 G 扩张因子为 c,则 G_f 亦然)。

② 长度为 t 的随机行走表示如下方式生成的 t 个顶点的序列。首先,从顶点集合中随机地选取一个作为起始顶点,对于 $i=2,\ldots,t$,第 i 个顶点是从 $i-1$ 个邻接顶点中随机选取的。我们强调指出的是,若一个顶点是自环的,则它也应被视作它本身的一个邻接顶点。

弱界的证明概略：用随机矩阵 M 标记在图 $G=(V,E)$ 上的随机步骤，令 ρ 表示 M 的第二大特征值的绝对值的界（最大的特征值是 1）。令 P 为 0-1"筛矩阵"，即仅在对角线上那些满足 $i\notin S$ 的位置 (i,i) 上值为 1。在 V 上用向量来描述（剩余）概率分布。对向量 $\vec{v}$，向量 $M\vec{v}$ 表示在图 G 上随机走一步由向量 $\vec{v}$ 分布推出的概率分布，（$P\vec{v}$ 表示通过将向量 $\vec{v}$ 中与 S 中的顶点对应的位置设为 0 推出的（剩余）分布。以向量 $\vec{\pi}$ 表示 V 上的均匀分布（每一个项的值等于 $1/|V|$），注意到 $M\vec{\pi}=\vec{\pi}$（因为均匀分布是与特征值 1 关联的特征向量）。

一个关键的结论是，随机 t 步行走不经过 S 的概率等于（非负）向量 $(PM)^{t-1}P\vec{\pi}=(PM)^t\vec{\pi}$ 中的元素的和。因为向量是非负的，因此我们能够计算它的 L_1-范数，而该范数受界于 $\sqrt{|V|}\cdot\|(PM)^t\vec{\pi}\|$，其中 $\|\cdot\|$ 表示欧几里得（Euclidean）范数（即 L_2-范数）。随后，我们将证明对任一向量 $\vec{z}$，$\|PM\vec{z}\|\leq((1-\mu)+\rho^2)^{1/2}\cdot\|\vec{z}\|$ 成立，推出：

$$\|(PM)^t\vec{\pi}\|_1\leq\{(1-\mu)+\rho^2\}^{t/2}\cdot\|\vec{\pi}\|=\{(1-\mu)+\rho^2\}^{t/2}\cdot\sqrt{|V|\cdot\frac{1}{|V|^2}}$$

且随机的 t 步行走不经过 S 的概率至多是 $((1-\mu)+\rho^2)^{t/2}$，对 $\mu\geq2\rho^2$（如 $\mu\geq1/2,\rho\leq1/2$）的情况，则上界变成 $(1-0.5\cdot\mu)^{t/2}$。

为了证明 $\|PM\vec{z}\|\leq\{(1-\mu)+\rho^2\}^{1/2}\cdot\|\vec{z}\|$，我们将 $\vec{z}$ 写成 $\vec{z}=\vec{z}_1+\vec{z}_2$ 的形式，$\vec{z}_1$ 是 $\vec{z}$ 在第一个特征向量方向（即 $\vec{\pi}$）上的分量，而 $\vec{z}_2$ 是与其正交的分量。利用 $M\vec{\pi}=\vec{\pi}$，$\|P\vec{\pi}\|=\sqrt{1-\mu}\cdot\|\vec{\pi}\|$，$\|M\vec{z}_2\|\leq\rho\cdot\|\vec{z}_2\|$ 以及 $\|P\vec{v}\|\leq\|\vec{v}\|$（对任一 $\vec{v}$），得到：

$$\begin{aligned}\|PM(\vec{z}_1+\vec{z}_2)\|&\leq\|PM\vec{z}_1\|+\|PM\vec{z}_2\|\\&\leq\sqrt{1-\mu}\cdot\|\vec{z}_1\|+\rho\cdot\|\vec{z}_2\|\\&\leq\sqrt{(1-\mu)+\rho^2}\cdot\sqrt{\|\vec{z}_1\|^2+\|\vec{z}_2\|^2}\\&=\{(1-\mu)+\rho^2\}^{1/2}\cdot\|\vec{z}_1+\vec{z}_2\|\end{aligned}$$

其中，最后一个不等式用到了柯西-许瓦茨（Cauchy-Schwarz）不等式（即 $\sum_i a_i\cdot b_i\leq(\sum_i a_i^2)^{1/2}\cdot(\sum_i b_i^2)^{1/2}$），最后一个等式用到了 $\vec{z}_1$ 与 $\vec{z}_2$ 的正交性。

我们在这里说明一下，引理中声明的下界可以被一般化为 $1-(1-\mu+\mu\cdot\rho)^t$，其中 ρ 是关于特征值比率的上界。■

算法　第二个引理（下面陈述）与定理 2.3.2 的证明实质（即简单放大）类似，但是两个证明之间有两个主要的不同之处。

1. 在定理 2.3.2 的证明中，我们使用了关于 $\{0,1\}^n$ 上的 k-序列数目的平凡组合（trivial combinatorial）陈述，每个 k-序列均在某个集合 S 中有元素（即，这样平均选取的 k-序列在集合 S 中没有元素的概率是 $(1-2^{-n}\cdot|S|)^k$）。这里我们对 S 的密度与经过 S 的确定类型的 k 序列的分数之间的关系做一个一般性的假设。即这里我们仅考虑在确定的正则图上行走 k 步生成的 k-序列。

2. 更为重要的是，定理 2.3.2 的证明借助于在（独立分布的）实例上对原始变换 f 求逆，而这里我们借助于在单个实例（讨论的序列是中间结果中的一个）上对 f 的逐次应用（以 g_σ-移动进行交叉）求逆。

因此下面的证明比定理 2.3.2 的证明要更加复杂。如下的引理将被使用到，且如早些时候组合参数设定的那样，$\beta(n+k(n)\cdot\log_2 d)=1-(1-0.5\cdot\alpha(n))^{k(n)/2}$。

引理 2.6.6(可约性引理):如同构造 2.6.3 那样,定义 d,$\{G_n=(\{0,1\}^n,E_n)\}$,$f:\{0,1\}^*\to\{0,1\}^*$,$k:\mathbb{N}\to\mathbb{N}$ 和 F_k。

- 令 $G_{f,n}\stackrel{\text{def}}{=}(\{0,1\}^n,E_{f,n})$,其中 $E_{f,n}\stackrel{\text{def}}{=}\{(u,v):(f(u),v)\in E_n\}$。
- 令 $\alpha,\alpha',\beta:\mathbb{N}\to[0,1]$和 $k:\mathbb{N}\to\mathbb{N}$ 满足 $\beta(n+k(n)\log_2 d)>\alpha(n)$且 $\alpha'(n)\geq\alpha(n)+2^{-n}$。

假设 $G_{f,n}$满足如下随机路径的性质:

- 对 $G_{f,n}$节点中的每一个度量 -$\alpha(n)$子集,长度为 $k(n)$的路径至少有 $\beta(n+k(n)\cdot\log_2 d)$ 的部分将经过 S 中的节点。

假设 f 是关于时间 $T(\cdot)$的 $\alpha'(\cdot)$-单向,则对每一个多项式时间可计算的 $\varepsilon:\mathbb{N}\to\mathbb{R}$,变换 F_k是关于时间 $T':\mathbb{N}\to\mathbb{N}$ 的$(1-\varepsilon(\cdot))\beta(\cdot)$-单向,其中 $T'(n+k(n)\cdot\log_2 d)\stackrel{\text{def}}{=}\frac{\varepsilon(n)^2\cdot\alpha(n^2)}{(K n+k(n))^3}\cdot T(n)$。

注意,此引理在 $\beta(n+k(n)\cdot\log_2 d)\leq\alpha(n)$的情况下没有意义。

略证:正如引理的名字所暗示的那样,证明是通过可还原参数进行的。这个参数与定理 2.3.2 的证明中使用到的参数类似。用反证法,我们假定对 $m\stackrel{\text{def}}{=}n+k(n)\log_2 d$,当输入 $F_k(u_m)$时置换 F_K 可在时间 $T'(\cdot)$内求出 $F_K(u_m)$的逆元成功的概率至少为

$$1-(1-\varepsilon(m))\cdot\beta(m)=1-\beta(m)-\varepsilon(m)\beta(m)$$

修改求逆算法,使之可对长度为 m 的输入的 $1-\beta(m)+\varepsilon'(m)$部分以压倒性的概率对 F_k 求逆,其中 $\varepsilon'(m)=\varepsilon(m)\beta(m)/2$(这点可以通过首先保证求逆算法至少以概率 $\varepsilon'(m)$对输入的 $1-\beta(m)+\varepsilon'(m)$部分求逆,再对如此的输入通过 $m/\varepsilon'(m)$次重复增加其成功概率而达到)。将运行时间为$(2m\cdot T'(m))/(\varepsilon(m)\beta(m))$的结果算法记为 A。注意 A 的输入对应于在图 G_n 上的 $k(n)$-长的路径。将 A 可以压倒性概率(即至少以概率 $1-2^{-n}$)对 $F_k(x,p)$求逆的路径(x,p)集合记为 I_n。

接下来,我们使用如下简记法:$k\stackrel{\text{def}}{=}k(n)$,$m\stackrel{\text{def}}{=}n+k\log_2 d$,$\varepsilon\stackrel{\text{def}}{=}\varepsilon(m)$,$\varepsilon'\stackrel{\text{def}}{=}\varepsilon'(m)$,$\beta\stackrel{\text{def}}{=}\beta(m)$,$\alpha\stackrel{\text{def}}{=}\alpha(n)$,$I\stackrel{\text{def}}{=}I_n$。记住$|I|\geq(1-\beta+\varepsilon')\cdot 2^m$。令 P_v是经过 v 的所有 k-长路径的集合,令 I_v为 I 的包含经过 v 的路径的子集(即 $I_v=I\cap P_v$)。若$|I_v|/|P_v|\geq\varepsilon'/k$,则定义 v 为好的(反之为坏的)。直观地,如果所有经过顶点 v 的路径至少在 ε'/k 部分可被 A 求逆,则称顶点 v 是好的。令 $I'=I\setminus\bigcup_{v\text{ bad}}I_v$;也就是说,$I'$包含所有的仅经过好的节点的"可求逆的"路径。明显地,我们得到如下结论。

结论 2.6.6.1:在所有路径集合中 I'的密度大于 $1-\beta$。

证明:记集合 S 在所有路径集合中的密度为 $\mu(S)=|S|/|P|$,则有

$$\begin{aligned}\mu(I')&=\mu(I)-\mu(\textstyle\bigcup_{v\text{ bad}}I_v)\\&\geq(1-\beta+\varepsilon')-\sum_{v\text{ bad}}\mu(I_v)\\&>1-\beta+\varepsilon'-\sum_{v}\frac{\varepsilon'}{k}\cdot\mu(P_v)\\&\geq 1-\beta\end{aligned}$$

其中,最后一个不等式是由于 P 中的每一条路径至多贡献 P_v中的 k。□

使用随机路径性质,我们得到如下结论。

结论 2.6.6.2:好的节点的密度大于 $1-\alpha$。

证明:反证法,令 S 是坏的节点的集合,并假设$|S|\leq\alpha\cdot 2^n$。由随机路径性质可得,因为 S 是度量(至少)α 的,则通过 S 中顶点的路径的部分至少为 β。也就是说,通过坏的顶点的路径

的部分至少为 β。但是 I' 不包含通过坏顶点的路径，因此 I' 至多能包含所有路径的 $1-\beta$，这与结论 2.6.6.1 矛盾。□

以下对 f 求逆的算法十分自然。算法使用用于对 F_k 求逆的标记为 A 的算法作为子程序。关于 y 对 f 求逆是这样完成的，将 y 放置在随机选择的一条路径 $\overline{p}$ 的一个随机点上，根据 $\overline{p}$ 的后缀随机地从 y 出发，并从 A 得到在 F_k 下的结果对的原像。

对 f 求逆的算法：输入 y，重复 $\frac{2nk}{\epsilon\beta}$ 次：

1. 从 $\{1, 2, ..., k\}$ 中随机选取 i，从 $\{1, 2, ..., d\}$ 中随机选择 $\sigma_1, \sigma_2, ..., \sigma_k$。
2. 计算 $y' = F(g_{\sigma_i}(y), \sigma_{i-1} ... \sigma_k)$。
3. 调用 A 得 $x' \leftarrow A(\sigma_1, \sigma_2, ..., \sigma_k, y')$。
4. 计算 $x = F(x', \sigma_1 ... \sigma_{i-1})$。
5. 若 $f(x) = y$，则停止，并输出 x.

求逆算法的分析(对好的 x)：因为 x 是好的，随机通过它的一条路径(按前述选择)对应于一条概率至少为 $\epsilon'/k = \epsilon\beta/2k$ 的"可逆路径"。如果这样一条好的路径已被选择，则我们可以压倒性的概率得到 $f(x)$ 的逆。对 f 求逆的算法重复该过程充分多的次数，以保证选择一条"可逆路径"时的压倒性的概率。

由结论 2.6.6.2 可得，好的 x 至少占全部 n 比特长字符串的 $1-\alpha$ 部分。因此，对输入 $f(U_n)$，我们求逆算法的成功概率至少是

$$(1-\alpha(n)) \cdot (1-2^{-n}) > 1-\alpha(n)-2^{-n} \geq 1-\alpha'(n)$$

求逆算法的运行时间是

$$\frac{2nk(n)}{\epsilon(m)\beta(m)} \cdot \frac{2m \cdot T'(m)}{\epsilon(m)\beta(m)} = \frac{4n\,mk(n)}{\epsilon(m)^2\beta(m)^2} \cdot T'(m) \leq T(n)$$

其中最后一个不等式用到了关系 $\beta(m) \geq \alpha(n)$，因此在时间 $T'(\cdot)$ 内至少以概率 $1-(1-\epsilon(\cdot))\beta(\cdot)$ 对 F_k 求逆的算法的存在，隐含地指出在时间 $T(\cdot)$ 内至少以概率 $1-\alpha'(\cdot)$ 对 f 求逆的算法的存在。而后者构成了引理假设的矛盾，因此引理成立。■

完成命题 2.6.4 的证明。将引理 2.6.5 应用于图 $G^l_{f,n}$，则有：对任一量度为 $\alpha(n)$ 的 $S \subseteq V$，在 $G^l_{f,n}$ 上长度为 t 的随机行走命中 S 的概率至少为 $1-(1-0.5 \cdot \alpha(n))^t$。因为 $G^l_{f,n}$ 中的边表示 $G_{f,n}$ 中 l-边的路径，因此在 $G_{f,n}$ 上 k-步行走所访问的顶点是对应 $G^l_{f,n}$ 上 (k/l)-步行走所访问的顶点的子集。在 $G_{f,n}$ 上 $k(n)$-步行走命中 S 的概率至少为 $1-(1-0.5 \cdot \alpha(n))^{k(n)/l}$。令 $\alpha'(n) = \alpha(n) + 2^{-n}$，$\beta(n + k(n) \cdot \log_2 d) = 1-(1-0.5 \cdot \alpha(n))^{k(n)/l}$，应用引理 2.6.6 的假设 f 是关于时间 $T(\cdot)$ 的 $\alpha'(\cdot)$-单向，则 F_k 是关于时间 $T'(\cdot)$ 的 $(1-\epsilon(\cdot))\beta(\cdot)$-单向，$\beta$ 与 T' 均在命题 2.6.4 中定义。这就完成了全部证明。

可选的分析。我们关于构造 2.6.3 的分析是通过使用扩张图的特征值比率进行的，而不是它们本身的组合数学定义(根据扩张性)。因为两个公式之间的变换不严格，所以我们无法根据扩张性描述我们的结论。因此，在更严格的分析中，我们用另一要求替换命题 2.6.4 中的条件 1：对某个 $\rho < 1$，可直接构造图族 $\{G_n\}$ 使其中的每个图的特征值比率最多是 ρ，然后用 $l \geq \max(1, \lceil \log_\rho(1/2) \rceil)$ 代替条件 4。修正后的命题的证明和原命题的证明类似，只是这里 $\{G_{l,n}\}$

的特征值比率小于或等于$\{G_n\}$的特征值比率。① 修正后的命题允许使用具有 18 度且特征值比率小于$\frac{1}{2}$的图直接构造图族$\{G_n\}$，反过来，这就允许我们设 $l=1$。于是，对 $k(n)=3n$ 和每个多项式时间可计算的$\varepsilon:\mathbb{N}\to\mathbb{R}$，置换 F_k 是关于时间 $T':\mathbb{N}\to\mathbb{N}$ 的$((1-\varepsilon(\cdot))\beta(\cdot))$-单向，其中

$$\beta(15n)\approx 1-\left(1-\frac{\alpha(n)}{2}\right)^{3n}$$

$$T'(15n)\approx\frac{(\varepsilon(n)\cdot\alpha(n))^2}{O(n)^3}\cdot T(n)$$

特别地，对 $\alpha(n)\leq 1/2n$，我们有 $\beta(15n)>1.02n\cdot\alpha(n)$，而对常量 α，有 $\beta(15n)>1-2^{-\Omega(n)}$。对本节开头提到的例子，设 $n=1000$，$k\approx 960$，可得：如果在实践中对长为 1000 的字符串 f 难于求逆的概率是 1%，那么对长为 5000 的字符串 F_k 难于求逆的概率是 99%。

2.7 其　　他

我们强调，前面提到的各种单向函数之间的相互关系是我们知道的仅有的结论。特别地：

- 弱单向函数（置换（有陷门的））能够被转化为强单向函数（置换（有陷门的））。反过来是不证自明的。
- 非均匀难处理问题隐含了均匀难处理问题，但反过来不一定成立。
- 陷门置换是单向置换的一种特殊情况，从而也是单向函数的特例。我们不知道是否任意的单向函数都能转化为单向置换或单向置换是否能转化为陷门置换②。已经提出了反面的证据（分别在[140]和[133]中，这里指出"黑盒子"变形不可能提供这样的转换）。
- 无爪函数对集合（置换）可生成单向函数集合（置换），但反过来还不知道。

2.7.1 历史记录

单向函数和陷门置换的概念源于 Diffie 和 Hellman 的基础论文[63]，Yao[210]介绍了弱单向函数。Rivest，Shamir 和 Adleman[191]介绍了 RSA 函数，而复数模的平方理论由 Rabin[187]提出并研究。还有一些作者提出建议，将单向函数建立在对随机线性码解码[29,108]以及子集和问题的难处理基础之上[132]。

弱的和强的单向函数存在的等价性（即定理 2.3.2）在 Yao 的文献里[210]，在[91]的第一个证明中已经暗示。2.6 节中对单向函数的有效放大来自于 Goldreich et al.[104]，而后者又用了来自[4]的技术工具（在[55,135]中也有介绍）。一般单向函数的存在性在 Levin 的文献

① 令 M 与引理 2.6.5 中的含义相同，令 R 表示映射 $v\mapsto f(v)$，如同对 M 的处理一样，考虑 MR 的第一个特征值与特征向量。更进一步，用 R 表示与均匀向量正交的子空间，则该子空间包含 MR 中其他的特征向量（这个子空间中的每一个向量都是 M 中的其他特征向量的线性组合）。令 $\vec{e}$ 是 MR 中与均匀向量正交的特征向量，ρ'是它的特征值。则 $\rho'\|\vec{e}\|=\|MR\vec{e}\|\leq\rho\cdot\|R\vec{e}\|\leq\rho\cdot\|\vec{e}\|$，其中对每一个随机矩阵 S，$\|S\vec{v}\|\leq\|\vec{v}\|$ 均成立（特别对矩阵 R 也成立）。

② 我们提出陷门函数（给定陷门，可求得一些原像）能由任意单向函数来构造（cf.[18]），但是构造函数的每一个像的原像数目都是指数数量的。

中陈述[150]。

核心断言的概念来源于 Blum 和 Micali 的文献[36]。他们还证明了只要“DLP 函数”是单向的，特定的断言就可以构成“DLP 函数”的核心(即在一个无限领域求幂)。随后，Yao 指出了如何将任意单向函数转化为核心断言(即，结果不是在[210]中陈述的，而是以口述的形式出现)。证明首先出现在 Levin 的文献[150]中(详见[114])。然而，Yao 的构造理论与定理 2.3.2 的证明过程中用到的构造理论相类似，几乎没有任何实用价值。

内积模 2 是任意单向函数(具有形式 $g(x,r)=(f(x),r)$)的核心这一事实已被 Goldreich 和 Levin[110]证明。本书里提出的证明实际来源于[5]的思想，分别由 Leonid Levin 和 Charles Rackoff 独立地提出。命题 2.5.4 中的改进来源于 Levin 的[151]。

定理 2.5.6(基于任意单向函数的有对数多位的核心函数)也是来源于[110]。计算异或引理(引理 2.5.8)来源于[208]，但其中的证明却来源于 Leonid Levin(在[117]中给出了核心函数的另一种可选择的构造方法)。

在[36,141,5,208]中已经暗示了特殊的置换类的核心断言(和函数)。特别地，Alexi et al.[5]证明了因数分解的难处理问题引出了由和数求模平方运算诱导的置换的核心断言。随后又找到了更简单更严密的证明[82]。

2.7.2 关于进一步阅读的建议

对 RSA 和 Rabin 函数的详细说明很少。特别是生成均匀分布的“已被鉴别的素数”和“素性(primality)校验”的计算问题更值得注意。Bach[9]提出了生成均匀分布素数和它相应的鉴别概率多项式时间算法。此算法提出的对素数 P 的鉴别由 P－1 的素因子分解和这些因子的素性鉴别组成。素性鉴别的递归形式来源于 Pratt 的证明[184]，其中的素数集合是属于 ***NP*** 问题的。然而，前述过程是不切实际的。相反，当我们实际应用 RSA(或 Rabin)函数时，我们宁愿随机产生一个整数，并用快速的素性校验子来检查它是否为素数，像[203,185]中提出的算法就可以。然而，这种算法并不能产生对素性的鉴定，并且可能以小参数概率证明一个合数是素数。给定一个素数，能够产生对素性鉴定的概率多项式时间算法(但不切实际)在[121,1]中提出。

认为 RSA，Rabin 和 DLP 函数是单向函数的常规思想是由于研究人员没有找到处理因数分解和离散对数的概率多项式时间算法(这种失败应该追溯到两个世纪以前还是仅仅几十年前是值得争论的)。对于目前所知道的处理因式分解和离散对数问题的最好算法的综述，读者可参阅 Odlyzko 所做的综述(分别在[178]和[179]中介绍)。

子集和问题在两种特定情况下很简单。在一种情况下，输入序列的构造是以一个简单的“隐藏序列”为基础的。例如，Merkle 和 Hellman[163]建议的子集和问题的一切实例的构造就是按如下步骤建立在一个“隐藏的超增长序列”的基础上的。设 $s_1,\ldots,s_n,s_{n+1}\stackrel{\text{def}}{=}M$ 是一个序列，满足对 $i=2,\ldots,n+1$ 都有 $s_i>\sum_{j=1}^{i-1}s_j$。我们称这样的序列为超增长序列。对 w 与 M 互素，考虑由 $(x_1,\ldots,x_n)$ 和 $\sum_{i\in I}x_i$ 组成的子集和问题实例，这里 $x_i\stackrel{\text{def}}{=}w\cdot s_i \bmod M$，$I\subseteq\{1,\ldots,n\}$。显然，$\omega$ 和 M 的信息使我们能够很容易地解决前述实例的子集和问题(简单检索超增长序列并对 $i=n,n-1,\ldots,1$，用迭代法决定是否有 $i\in I$)。我们的希望是如果不给出 ω 和 M，解决子集和问题就会很困难(即使是在超增长序列基础上生成的实例也很困难)(这就

会导致陷门单向函数)。不幸的是,希望还没有实现。Shamir 提出一个解决拥有隐藏超增长序列的子集和问题实例的有效算法[197]。另一种子集和问题易解决的情况是低密度实例。在这些实例当中,元素的二进制表示的长度远远大于元素数目(即 $|x_1|=\cdots=|x_n|=(1+\varepsilon)n$,对某一常数 $\varepsilon>0$)。有关进一步的详细讨论,参照 Lagarias 和 Odlyzko 的文章[145]以及 Brickell 和 Odlyzko 后面的综述[43]。

看起来与子集的问题相关的两类计算问题是随机线性码的译码问题和在整数点阵里找最接近的向量问题。在所有这三种问题中,都是找一个给定元素的线性组合使得它们的和等于或最靠近目标值。然而,这种相似性是表面的,因为三种情况的算法是不同的。在子集和问题中,我们强调整数的加法;在线性码问题中,我们进行有限域上向量空间内的加法(主要是两个向量的加法);在整数点阵问题中,进行的是实向量的加法(或有理数或整数的加法)。我们指出,随机线性码的译码问题是编码理论上长期存在的未解决的问题[207]。关于点阵问题的复杂性,在理论上限[148]和实际操作[195]中似乎存在很大的误差。

由 Ajtai[3](cf. [101])给读者指出一个很好的结果:如果关于整数点阵的特定的计算问题在最坏的情况下是难处理的,那么单向函数就存在。这个结果在将可能出现的最坏情况下的难题转化为一般情况下的难题的过程中是独特的。

观察 2.5 节提出的将单向函数转化为核心断言的一般的有效转化,我们不能提出证据来证明特定的一般的断言是单向函数的特定的常用替代的核心。RSA 和 Rabin 函数的核心断言在[82;cf. 5]中详细介绍,各种"DLP 函数"的核心断言的详细介绍在[141;cf. 36]中。

归功于 Yao 提供了基于任意单向函数的核心断言的存在性的证明。所谓的证明按如下两步进行。首先证明适中形式的核心断言的存在性;特别地,给定一个单向函数 f,我们可以构造一个单向函数 f' 和一个多项式时间可计算的断言 b',使得任意多项式时间预言机在给定 $f'(U_n)$ 时,至少有 $\frac{1}{2n}$ 的概率猜不中 $b'(U_n)$(例如,设 $f'(x,i)=(f(x),i)$,且 $b'(x,i)$ 是 x 的第 i 位)。第二步,也是最重要的一步,被称为 *Yao* 氏异或引理,是要证明选取许多这样的"适中核心断言"并对它们集体求异或就可以得到一个核心断言。也就是说,对 $t=|w_1|^2=\ldots=|w_t|^2$,我们令 $b''(w_1,\ldots,w_t)=\bigoplus_{i=1}^{t}b'(w_i)$,$f''(w_1,\ldots,w_t)=(f'(w_1),\ldots f'(w_t))$,并证明 b'' 是 f'' 的核心。Yao 氏异或引理的复杂性理论中找到了其他方面的应用[114,134]。

由 Levin 提出的一般情况下的复杂性理论[149]与单向函数的概念有一定程度的关联,这方面理论的综述已经存在[24,96]。简单地说,本书和 Levin 文献的不同之处是,本书中的难处理实例(一般情况下)能够由这些实例的"发生器"很容易地解决,而后者中即使用"发生器",这些实例也是很难解决的。然而,Levin 介绍的一般情况下的可约性概念与本书是有关的。

有关扩展图和随机路径的进一步讨论来自[6,167]。特别地,引理 2.6.5 是 Kahale 的 Corollary 6.1 的一个特例。扩展图的明确构造已经出版[85,154],同时在 2.6 节末尾还介绍了一种特定的构造方法[154]。

2.7.3 未决问题

如 2.1 节所提,$\mathcal{NP}\setminus\mathcal{BPP}\neq\emptyset$ 是单向函数存在的一个必要条件。然而,我们还不知道 $\mathcal{NP}\setminus\mathcal{BPP}\neq\emptyset$ 是否隐含一些实际结果(即难处理实例确实存在,但却在简单分布中很少出现)。指出 $\mathcal{NP}\setminus\mathcal{BPP}\neq\emptyset$ 的任何程序都隐含了一般情况下难处理实例的某些形式,而后者又指出单向

函数的存在意义重大。

转而考虑比较低一些的目标，我们提出两个属于本章提出问题类型的延伸结论的未决问题。我们相信对这些难题的任意一种解决方案都要求我们发现新的重要的范例。首先，在有效扩充单向置换（在 2.6 节中介绍）的续篇中，我们寻找一个类似的变形，使它适用于任意（弱）单向函数。目前，我们只知道对特殊类型的函数（即正则函数[104]）有这样的变形。我们认为，为任意单向函数提供一个有效的扩充是一个非常重要的未决问题。这对基于任意单向函数的伪随机数发生器的更有效的构造也是很有用的（详见 3.5 节）。

一个更尖锐的具有实际重要性的未决问题是为 RSA 和 Rabin 函数提供定义域更广的核心函数。特别地，假设平方模 N 是单向的，那么返回 x 的前半部分位的函数是否为平方模 N 的核心？在[130]中提供了一些肯定回答的依据。肯定的回答使我们能够构造非常有效的伪随机发生器和建立在因数分解难处理问题的假设基础上的公钥加密算法。

2.7.4　习题

习题 1：缩小动机讨论和单向函数定义之间的差距。如果存在一个满足以下两个条件的概率多项式时间算法 G，我们就说函数 $h:\{0,1\}^*\rightarrow\{0,1\}^*$ 在大多数情况下困难，但有辅助输入时就变得容易了：

1. 存在一个多项式时间算法 A，使得对每个在 G 内变化的 (x,y)（即每个 (x,y) 都是 $G(1^n)$ 对某一输入 1^n 的可能输出），有 $A=(x,y)=h(x)$；

2. 对每个概率多项式时间算法 A'、每个正多项式 $p(\cdot)$ 和每个足够大的 n，有

$$\Pr[A'(X_n)=h(X_n)]<\frac{1}{p(n)}$$

其中 $(X_n,Y_n)\stackrel{\text{def}}{=}G(1^n)$ 是一个表示 G 的输出的随机变量。

试证明：如果“多数情况困难但有辅助输入时容易”的函数存在，那么单向函数存在。

提示：定义一个函数，使之把 G 使用的硬币映到它的第一个输出上。

习题 2：单向函数和 $\boldsymbol{P}$ 对 $\boldsymbol{NP}$ 问题（第一部分）。试证明：单向函数存在意味着 $\boldsymbol{P}\neq\boldsymbol{NP}$。

提示：对每个多项式时间可计算的函数 f，定义集合 $L_f\in\boldsymbol{NP}$，若使得 $L_f\in\boldsymbol{P}$ 则存在一个对 f 求逆的多项式时间算法。

习题 3：单向函数和 $\boldsymbol{P}$ 对 $\boldsymbol{NP}$ 问题（第二部分）。假设 $\boldsymbol{P}\neq\boldsymbol{NP}$，构造一个函数 f，满足下面 3 个声明：

1. 函数 f 的多项式时间是可计算的。

2. 不存在总是能对 f 求逆（即对每个在 f 的值域中的 y 来说，都能成功地对 f 求逆）的多项式时间算法。

3. 函数 f 不是单向的。而且，存在一个可以指数小的失败概率对 f 求逆的多项式时间算法，其概率空间（通常情况下）是在所有可能的输入选择（即 $f(x)$）和此算法的内部硬币投掷上均匀分布的。

提示：按如下要求定义一个函数 f_{sat}：如果 τ 满足指定的公式 ϕ，则 $f_{\text{sat}}(\phi,\tau)=(\phi,1)$，反之，$f_{\text{sat}}(\phi,\tau)=(\phi,0)$。修改这个函数使之在大多数场合下易于求逆，这样对 f_{sat} 求逆就可以转化为对修改后的函数求逆。（提示：对 f' 的作用域来说，修改后的函数 f' 同 f_{sat} 一致，只占可忽略

不计的一部分，其余的部分都易于求逆。）

习题 4：假设 f 是单向函数，对某一函数 $l:N\to N$ 如下条件成立：

1. 对所有 x，都有 $|f(x)|=l(|x|)$；

2. $l(n)=l(m)$ 仅当 $n=m$（即 l 是 1-1 函数）；

3. 对所有 n，都有 $l(n)\geq n$。

请证明：若给定 $f(x)$，则能在多项式时间 $|x|$ 内产生 $1^{|x|}$。

提示：上述条件能保证 $|x|\leq|f(x)|$，并且 $|x|$ 可以被 $|f(x)|=|f(1^{|x|})|$ 惟一确定。

习题 5：设 f 是一个强单向函数，试证明：对每个概率多项式时间算法 A 和每个正多项式 $P(\cdot)$，集合

$$B_{A,P}\stackrel{\text{def}}{=}\{x:\Pr[A(f(x))\in f^{-1}(f(x))]\geq\frac{1}{p(|x|)}\}$$

在所有字符串构成的集合中具有可忽略不计的密度（即对每个多项式 $q(\cdot)$ 和所有足够大的 n，$\frac{|B_{A,P}\cap\{0,1\}^n|}{2^n}<\frac{1}{q(n)}$ 成立）。

习题 6：非均匀单向函数的另一定义：通过允许定义 2.2.6 中的电路是概率的（即含有一个均匀分布的辅助输入），可得到新的定义。试证明这个新定义等价于原定义。

习题 7：加法易于求逆：我们按某种固有的方式把比特串和正整数关联起来（例如，n 比特长的字符串 $\sigma_{n-1}\cdots\sigma_0$ 与整数 $2^n+\sum_{i=0}^{n-1}\sigma_i\cdot 2^i$）相关联：

1. 定义 $f_{add}:\{0,1\}^*\to\{0,1\}^*$，使得 $f_{add}(xy)=x+y$，其中 $|x|=|y|$。试证明 f_{add} 不是单向函数（甚至连弱单向函数都不是）。

2. 重定义 $f_{add}:\{0,1\}^*\to\{0,1\}^*$，使得 $f_{add}(xy)=\text{prime}(x)+\text{prime}(y)$，其中 $|x|=|y|$，prime(z) 是大于 z 的最小素数。试证明 f_{add} 不是单向函数。

作为预备，可以证明 $f_{XOR}(xy)=x\oplus y$（其中 $|x|=|y|$）不是单向函数。

提示（第二部分）：请不要试图利用 Prime(N) 非常大（例如，大于 $N+\text{poly}(\log N)$）的可能性。这样一个（数论上的）结论不可能得到证明。而且，一般认为，存在这样一个常数 c，使得对所有整数 $N\geq 2$，有 $\text{prime}(N)<N+(\log_2 N)^c$。因此，$f_{add}$ 的多项式时间是可计算的这一结论是可能的。于是可以证明它易于求逆。

习题 8：基于因数分解的困难性的单向函数：在本题中，假设对这样的合数分解因数是不可行的，此合数是两个多项式相应长度的素数的乘积。也就是说，对每个概率多项式时间算法 A、每个正多项式 p、所有足够大的 n 以及每个 $\sqrt{n}<m<n^2$，有

$$\Pr[A(P_m\cdot Q_n)=P_m]<\frac{1}{p(n)}$$

其中 P_m 和 Q_n 分别是长为 m 和 n 的独立均匀分布的素数（考虑素数密度理论，其保证 n 比特的素数中至少有 $1/n$ 是素数[7]）。

1. 令 $f_{mult}(x,y)=x\cdot y$，其中 $|x|=|y|$。

(a)（较易）证明：f_{mult} 是弱单向函数。

(b)（较难）证明：f_{mult} 是强单向函数。

提示：首先利用这样一个事实：均匀地选择一个 n 比特长的整数，考虑所有它的小于 $2^{\sqrt{n}}$ 的素因数的积，这个积小于 $2^{n/3}$ 的概率压倒多数。接着证明：如果 f_{mult} 可以以不可忽略的概率求逆，那么两部分原像有大小至少是 $2^{\sqrt{n}}$ 的素因数的概率也是不可忽略的。这个观点可运用简化

证明。(前面使用的数论结论可利用著名的关于平滑数的分布结论来证明;后者可参见[47]。)

2. 令 $f_{\text{mult}}(x_1,\ldots,x_{n^2})=\prod_{i=1}^{n^2}$,其中对所有 I 有 $|x_i|=n$。试证明 f_{mult} 是强单向函数。

提示:为了对两个 n 比特的素数的乘积进行因数分解,应证明如何利用一个算法以不可忽略不计的概率对 f_{mult} 求逆。请记住,必须给前一算法输入一分布函数,这可作为前提条件(或充分接近于它)。

习题 9(Bao Feng 推荐的):反驳如下猜想:

对每个(等长的)单向函数 f,函数 $f'(x)\stackrel{\text{def}}{=}f(x)\oplus x$ 也是单向的。

提示:令 g 是一个(等长的)单向函数,考虑定义在相同长度的字符串对上的函数 f,有 $f(y,z)\stackrel{\text{def}}{=}(g(y)\oplus z,z)$。

习题 10:试证明单向函数的值域不可能是多项式大小的。也即,证明:如果 f 是(弱)单向函数,那么对每个多项式 $p(\cdot)$ 和所有足够大的 n,有 $|\{f(x):x\in\{0,1\}^n\}|>p(n)$ 成立。

提示:假设 $|\{f(x):x\in\{0,1\}^n\}|\leq p(n)$。为了使输入 $y=f(U_n)$ 时,对 f 求逆成功的概率达到 $1/p(n)$,必须均匀地选择 $r\in\{0,1\}^n$ 并期望 $f(r)=y$。为了使输入 $y=f(U_n)$ 时,对 f 求逆成功的概率为 $1-\varepsilon(n)$,我们均匀地选择多个这样的 r_i,并期望 y 是"大量的",且所有"大量的"f 的像可通过某一 $f(r_i)$ 找到。(提示:如果 $\Pr[f(U_n)=y']\geq\frac{\varepsilon(n)}{2p(n)}$,那么 y'是大量的。)

习题 11:试证明等长的单向函数的周期不可能是多项式约束定的。也就是说,对每个函数 f,定义 $cyc_f(x)$,为满足 $f^i(x)=x$ 的最小正整数 i,其中 $f^{i+1}(x)=f(f^i(x))$,$f^0(x)=x$。试证明,如果 f 是(弱)单向函数,那么对每个多项式 $p(\cdot)$和所有足够大的 n,$cyc_f(U_n)$的期望值大于 $p(n)$,其中 U_n 是一个在$\{0,1\}^n$ 上均匀分布的随机变量。

提示:请注意,如果 $\mathrm{E}cyc_f(U_n)>p(n)$,那么对每个多项式 q,$\Pr[\mathrm{cyc}_f(U_n)>q(n)\cdot p(n)]<1/q(n)$成立。为什么需要等长这一条件?

习题 12:假设单向函数(或置换)存在,构造没有亚指数周期的单向函数(或置换)。也就是说,令 $\mathrm{cyc}_f(x)$ 和习题 11 中定义一样,那么对所有的 x,构造的 f 必须满足 $cyc_f(x)\geq 2^{|x|/2}$。

提示:给定一个单向函数(或置换)f',对合适的 h 和 $|x'|=|x''|$ 构造 $f(x',x'')\stackrel{\text{def}}{=}(f'(x'),h(x''))$。什么是合适的 h?

习题 13:含"固定点"的单向函数:试证明如果单向函数存在,则存在这样一个单向函数 f,对所有 n 都有 $f(0^n)=0^n$。同理可证单向置换。

提示:第一部分显然。对第二部分,使用任意单向置换 f',如果 $y\in\{0,1\}^{|x|}\setminus\{0\}^{|x|}$,则 $f(x,y)=(f'(x),y)$,反之,$f(x,0^{|x|})=(x,0^{|x|})$。

习题 14:设$\{(a_n,b_n):n\in\mathbb{N}\}$在(确定性的)多项式时间内是可识别的,其中 $a_n,b_n\in\{0,1\}^n$。试证明,如果单向函数存在,那么存在一个单向函数 f,对所有 n 都有 $f(a_n)=b_n$。同理可证单向置换。

提示:第一部分显然。对第二部分,考虑任意单向置换 f',并假设 $f'(a_n)\neq b_n$。再构造一个单向置换 f,以满足交换 f'的两个值的需要。

习题 15:关于强化定理 2.3.2 的不可能性(第一部分):假设弱单向函数的定义进一步减弱,乃至要求每个概率多项式时间算法不能对函数求逆的概率是可忽略不计的。也就是说,定义 2.2.2 中的量词顺序颠倒了(我们用"对每个算法,存在一个多项式"代替"存在一个多项式,使得每个算法")。试证明把定理 2.3.2 的证明扩展到这种情况的困难性。

提示:假设存在一个算法族,每个算法有一个多项式 $p(\cdot)$,使得在时间 $p(n)$ 约束的算法不能对函数求逆的概率是 $1/p(n)$。再证明这样的族其实是不存在的(即用反证法)。

习题 16:关于强化定理 2.3.2 的不可能性(第二部分)(因 Steven Rudich 而得):假设强单向函数的定义进一步加强,乃至要求每个概率多项式时间算法不能对函数求逆的概率是某一特定的可忽略不计的值(例如,$2^{-\sqrt{n}}$)。试证明把定理 2.3.2 的证明扩展到这种情况的困难性。

提示:假设我们像原先的证明那样构造一个强单向函数 g。接着假设存在一个求逆算法 A,当输入 $g(U_n)$ 时,A 能对 g 求逆的概率是 $\varepsilon(n)$。然后证明任何对弱单向函数 f(此函数把算法 A 当作黑盒子)求逆的算法至少要调用它 $\frac{1}{\mathrm{poly}(n)\cdot\varepsilon(n)}$ 次。

习题 17:高深课题:分布式单向函数[131]:如果存在这样一个正多项式 p,对每个多项式时间算法 A 和所有足够大的 n,$(U_n,f(U_n))$ 和 $(A(1^n,f(U_n)),f(U_n))$ 之间的统计差异大于 $1/p(n)$。(也就是说,求逆的目标就是要给出一个均匀分布的原像而不是任意原像,并根据 A 的输出偏离此分布的程度来衡量是否失败。)

1. 试证明,如果 f 是弱单向函数(如定义 2.2.2),那么它就是分布式单向函数。
2. 试证明,如果分布式单向函数存在,那么单向函数也存在。

提示(第 2 部分):像 3.5 节那样利用哈希函数散射思想。特别地,给定一个分布式单向函数 f,考虑函数 $F(x,i,h)=(f(x),h_i(x),i,h)$,其中 $x\in\{0,1\}^n$,$i\in\{1,\ldots,n\}$,$h:\{0,1\}^n\to\{0,1\}^n$ 是一个哈希函数,$h_i(x)$ 是 $h(x)$ 的 i 比特长的前缀。试证明 F 是弱单向函数。

提示(第 2 部分,额外帮助):相反,假设对输入 (x,i,h) 部分来说,至少有 $1-\varepsilon(n)>1-(2n)^{-1}$ 对 F 可以求逆,其中 $|x|=n$。那么对任意 $l:\mathbb{N}\to\mathbb{N}$,输入 $(x,\lfloor\log_2|f^{-1}(f(x))|\rfloor+l(n),h)$ 时,至少有 $1-\varepsilon(n)$ 可以对 F 求逆。若 $y=f(x)$,则我们可以按如下方式产生 y 在 f 作用下的随机原像。首先,对 $l(n)=O(\log n)$,我们寻找这样一个 i,使得 $i=\lfloor\log_2|f^{-1}(f(x))|\rfloor+l(n)\pm O(1)$(当输入 (y,i,r,h) 时,可以这样做来对 F 求逆,其中 h 和 $r\in\{0,1\}^i$ 是均匀选择的,并且如果找得到原像的概率是 $2^{-l(n)}$,就选择 i)。接着,利用这个 i,输出 (y,i,r,h) 在 F 作用下的原像,其中 h 和 $r\in\{0,1\}^i$ 是均匀选择的(求逆失败时,我们将重试)。然后证明此算法的输出分布偏离期望的分布的概率至少是 $O(2^{l(n)}+2^{2l(n)+2\log_2 n}\cdot\varepsilon(n))$,因此结论成立。

习题 18:单向函数和单向函数类:

1. 给定任意单向函数类 (I,D,F),把它表示成单个单向函数。
2. 给定任意单向函数 f,把它表示成一个单向函数类(注意:这个方向是十分简单的)。

习题 19:有关单向函数类的规定:试证明,为了不失一般性,可以修改(单向)函数类中的算法 I 和 D,使得它们都使用输入长度那么多的硬币。

提示:利用填料(padding)。

习题 20:论证有关单向函数类的规定:试证明,给出函数求逆算法的索引对定义单向函数类来说是至关重要的。

提示:考虑函数类 $\{f_i:\{0,1\}^{|i|}\to\{0,1\}^{|i|}\}$,其中 $f_i(x)=x\oplus i$。

习题 21:*Rabin* 函数类和因数分解:试证明,*Rabin* 函数类是单向的当且仅当整数(该整数是两个等长的二元扩展素数的乘积)的因数分解是严格意义上的难处理。

提示:参考附录 A。

习题 22:无爪函数隐含单向函数:令 (I,D,F) 为无爪单向函数类(参见定义 2.4.5)。试证明,对每个 $\sigma\in\{0,1\}$,三个算法 (I,D,F_σ)(其中,$F_\sigma(i,x)\overset{\mathrm{def}}{=}F(\sigma,i,x)$)构成一个强单向函数类。

用“置换”代替“函数”,重复此习题。

习题 23:更多有关把图的同行异构作为基本单向函数的不充分性的讨论:继续 2.4.6 小节中的讨论,为了把单向函数建立在对图的同行异构问题的困难性的猜想上,我们考虑另一个建议。这里我们提出一个由三个算法(I_{GI}, D_{GI}, F_{GI})定义的函数类。输入 1^n 时,算法 I_{GI} 在 n 个顶点上均匀地选择一个 $d(n)$-正则图(即图中 n 个顶点的度都是 $d(n)$)。当输入一个有 n 个顶点的图时,算法 D_{GI} 在 n 元对称群上随机选择一个置换(即 n 元置换集)。当输入一个(n 边)图 G 和(n 元)置换 π 时,算法 F_{GI} 返回 $f_G(\pi) \stackrel{\text{def}}{=} \pi G$。

1. 给出 I_{GI} 的多项式时间实现方法。

2. 若已知图的同行异构问题的算法,$d(n)$的哪些值应该明确避免?

3. 利用已知算法,证明无论使用哪个函数 $d(\cdot)$,上述函数类都不具有单向性。

提示:第 2 和第 3 项需要研究相关文献。特别地,对 $d(n)$ 的某些值来说,存在一个判断同行异构的多项式时间算法。而且,要证明 3,必须有一个在随机选择的 d-正则图对上高速运行的算法。

习题 24:假设单向函数存在,试证明:存在这样一个单向函数 f,使得原像没有一个比特组成核心断言。

提示:给定单向函数 f,构造一个函数 g,使得 $g(x,I) \stackrel{\text{def}}{=} (f(x_I), x_{\bar{I}}, I)$,其中 $I \subseteq \{1,2,\cdots |x|\}$,$x_S$ 表示只取集合 S 中 x 比特位(即 $x_{\{i_1,\ldots,i_s\}} \stackrel{\text{def}}{=} x_{i_1} \cdots x_{i_s}$,其中 $x = x_1 \cdots x_{|x|}$)。那么我们如何才能预测每个比特?为了获得更多的“引人关注的”可预测性,考虑 $g(x, I_1, \cdots, I_t) \stackrel{\text{def}}{=} (f(x_{\bigcap_{j=1}^{t} I_j}), x_{\bigcup_{j=1}^{t} I_j}, I_1, \ldots, I_t)$。(作为函数 $|x|$)t 应该取何值?

习题 25:1-1 函数的核心断言意味着此函数是单向的:设 f 是一个 1-1 函数(为了简化,可以假设函数是等长的),假设 b 是 f 的核心。

1. 试证明,如果 f 是多项式时间可计算的,那么它就是强单向的。

2. 试证明,(无论 f 是否是多项式时间计算的)函数 f 必须至少是“弱难于求逆”;也就是说,对某一正多项式 p 和所有足够大的 n,每个概率多项式时间算法 A 必须满足 $\Pr[A(f(U_n)) \neq U_n] > 1/p(n)$。进一步证明,对某一正多项式 p 和所有足够大的 n,每个概率多项式时间算法 A 必须满足 $\Pr[A(f(U_n)) = U_n] < \frac{1}{2} + \frac{1}{p(n)}$。

提示:利用求逆算法预测核心。注意区分可以核实求逆算法的正确性的情况(即第 1 条)和不能核实的情况(即第 2 条)。

习题 26:无偏核心断言(由 Erez Petrank 提出):假设单向函数存在,试证明(此函数的)无偏核心断言(即谓词 b 满足 $\Pr[b(U_n) = 1] = \frac{1}{2}$)也存在。

提示:对定理 2.5.2 定义的谓词稍作修改(即只能在 x 全零时修改它)。或者,把某一函数 f 的核心 b 转化为函数 $f'(x,\sigma) = (f(x),\sigma)$ 的核心 $b'(x,\sigma) = \sigma \oplus b(x)$。

习题 27:通用核心断言:如果对每个单向函数 f 都有谓词 b 是 f 的核心,那么此多项式时间可计算的谓词 $b:\{0,1\}^* \rightarrow \{0,1\}$ 被称为通用核心断言。需要注意的是,定理 2.5.2 提出的谓词是“几乎通用”的(即对每个单向函数 f,此谓词是 $f'(x,r) = (f(x),r)$ 的核心,其中 $|x| = |r|$)。试证明通用核心断言不存在。

提示:设 b 是参考通用核心断言,f 是任意单向函数,可定义单向函数 $f'(x) = (f(x), b(x))$。

习题 28：对定理 2.5.2 的另一种陈述（由 Russell Impagliazzo，Madhu Sudan 和 Luca Trevisan 提出）：定理 2.5.2 提出的核心断言可以视作等于 x 的哈达马代码中的第 i 比特的 $b(x,i)$，其中哈达马代码是冗余度最大的（非重复）线性码（即字符串 $x\in\{0,1\}^n$ 映射到所有可能的 2^n 种其比特的线性组合。定理 2.5.2 用到的证明方法实际上为哈达马代码提供了一个"列表-解码"算法。特别地，若给出接近于字符串 $y\in\{0,1\}^{2^n}$ 的比特的预言和参数 $\varepsilon>0$，则在 $\mathrm{poly}(n/\varepsilon)$ 时间内，我们可以重获所有字符串 $x\in\{0,1\}^n$，使得 $H(x)$ 和 y 最多有 $(\frac{1}{2}-\varepsilon)\cdot 2^n$ 个位置不同。

1. 验证上述声明，即证明具有上述特性的"列表-解码"算法（用于哈达马代码）隐含在定理 2.5.2 的证明中。

2. 设 C 是从 n 比特字符串映射到 $l(n)$ 比特字符串上的纠错代码。C 应该满足什么要求才能使 $b(x,i)$（定义为 $C(x)$ 的第 i 比特）构成函数 $f'(x,i)=(f(x),i)$（对每个单向函数 f 来说）的核心断言。

提示：请注意我们必须支持任何形如 $1/\mathrm{poly}(n)$ 的 ε，且要记住 b 是多项式时间可计算的。还要注意 $|i|=\mathrm{poly}(|x|)$。为什么？

3. 利用 Reed-Solomon 代码的列表-解码算法[203]，提出这样一个核心预言。特别地，对任意"好的"超对数函数 $l:\mathbb{N}\to\mathbb{N}$（如 $l(n)=(\log_2 n)^2$），有 $|i|=l(|x|)$。

习题 29：和习题 28 的最后一条相反，试证明，如果 $b(x,y)$ 是每个形如 $f'(x,y)=(f(x),y)$ 的单向函数的核心，那么 $|y|$ 一定大于 $|x|$ 的对数。

提示：若 $|y|=O(\log|x|)$，则 $y=0^{|y|}$ 发生的概率是 $1/\mathrm{poly}(|x|)$。利用这一结论扩展习题 27 的证明。

习题 30：简化定理 2.5.2 的证明：假设给定一个接近于任意谓词 $P_x:\{0,1\}^{|x|}\to\{0,1\}$ 的预言，满足

$$\Pr[P_x(U_{|x|})=b(x,U_{|x|})]\geq\frac{1}{2}+\varepsilon$$

1. 给出一个概率预言机，它能运行 $\mathrm{poly}(|x|)/\varepsilon^2$ 步，且若给定任意接近于这样的 P_x 的预言，输出的字符串列表中至少有 $\frac{1}{2}$ 的概率包含 x。

2. 令 M 为一个预言机，对任意预言 P_x，和以前一样输出一系列字符串，其中至少有 $\frac{1}{2}$ 的概率包含 x。试证明 M 必须走 $\min(2^{\Omega(n)},\Omega(n/\varepsilon^2))$ 步。

提示（第 1 部分）：设 $n=|x|$，假设 $\varepsilon\geq 2^{-n}$，定理 2.5.2 的证明隐含着一个机器，它运行 $\mathrm{poly}(n)/\varepsilon^2$ 步并输出单个字符串，且等于 x 的概率至少是 $\varepsilon^2/\mathrm{poly}(n)$。这就引发了一个在多项式时间 $\mathrm{poly}(n)/\varepsilon^4$ 内运行并输出期望的列表的机器。在时间 $\mathrm{poly}(n)/\varepsilon^2$ 内运行并输出期望的列表的机器隐含于命题 2.5.4 的证明中。

提示（第 2 部分）：考虑按如下方式定义的概率预言（或程序）。首先，在 $\{0,1\}^n$ 上均匀地选择 x，程序的剩余部分 x 固定。接着，每发一个请求 q（不管是否是第一次），预言响应 $b(x,q)$ 的概率是 $\frac{1}{2}+\varepsilon$，反之响应 $1-b(x,q)$。试证明每次请求获得的有关 x 的信息量是 $O(\varepsilon^2)$。另一方面，包含 x 的 l 字符串列表至少包含 $n-\log_2 l$ 比特关于 x 的信息量。最后利用 M 的输出列表的长度和 M 的请求数目都受 M 的运行时间约束这一明显的事实。

习题 31：核心函数的另一种定义：令 $h:\{0,1\}^*\to\{0,1\}^*$ 和 $I:\mathbb{N}\to\mathbb{N}$ 对所有 $x\in\{0,1\}^*$ 都有 $|h(x)|=I(|x|)$。如果对每个多项式时间算法 A、每个正多项式 p 和所有足够大的 n，下

式成立，那么我们就说 h 近似 f 是困难的。

$$\Pr[A(f(X_n)) = h(X_n)] < 2^{-l(n)} + \frac{1}{p(n)} \tag{2.16}$$

其中 X_n 在 $\{0,1\}^n$ 上均匀分布。

1. 试证明，对满足 $l(n)=O(\log n)$ 的 $l:\mathbb{N}\to\mathbb{N}$ 和多项式时间可计算的 $h:\{0,1\}^*\to\{0,1\}^*$，函数 h 是 f 的核心当且仅当 h 近似 f 是困难的。

2. 证明通常情况下第 1 部分的一个方向并不成立(即对超对数增长的 l 来说)。

评论：本习题和 3.3.5 小节有关。

提示(主要对第 1 部分)：假设存在一个违反等式 2.16 的算法 A，如同定义 2.5.5 构造一个算法 D'，使得 $D'(y,\alpha)=1$ 当且仅当 $A(y)=\alpha$。试证明 D' 的差距困难至少是 $S(n)-2^{-l(n)}$，其中 $S(\cdot)$ 表示 A 成功的概率。另一方面，假设存在一个违反定义 2.5.5 条件的算法 D'，可以构造一个违反等式 2.16 的算法 A。特别地，为了不失一般性，假设 $\Pr[D'(f(X_n), h(X_n))=1]=\Pr[D'(f(X_n), R_{l(n)})=1]+\varepsilon(n)$，其中 $\varepsilon(n)>\frac{1}{p(n)}$。那么当输入 y 时，算法 A 均匀地选择 $r\in\{0,1\}^{l(n)}$ 和 $r'\in(\{0,1\}^{l(n)}\setminus\{r\})$，调用 D'，若 $D'(y,r)=1$，则输出 r，否则输出 r'。然后证明 D' 成功的概率至少是 $\frac{\varepsilon(n)}{2^{l(n)}-1}$。

第 3 章

伪随机发生器

本章我们讨论伪随机发生器。不太严格地讲，它们是一些有效的确定性程序，可以将短的随机挑选的种子扩展为长的“伪随机”比特序列（见图 3.1）。伪随机序列定义为计算不可分辨（computational indistinguishability）序列，即通过有效算法不能将它们从真正的随机序列中分辨出来。因而计算不可分辨（即不能通过有效算法分辨）的概念不但在我们的讨论中起着关键作用，而且在后续的章节中也起着非常关键的作用，特别是在安全加密、零知识证明和密码协议中。

伪随机理论也适用于函数，因而有了伪随机函数的概念，它在密码学应用中是一个很有用的工具。

除了伪随机分布、伪随机发生器和伪随机函数的定义外，本章也包含基于各种不同单向函数的伪随机发生器（和伪随机函数）的构造。特别地，基于单向置换的存在性而构造了非常简单而有效的伪随机发生器。我们着重介绍混合技术，它在许多证明中起着重要的作用（该技术的首次使用和进一步讨论参见 3.2.3 小节）。

内容安排 3.1～3.4 节讲的是关于伪随机发生器的基本讨论、定义和构造：首先给出一个启发性讨论（见 3.1 节），接下来是计算不可分辨性的一般定义（见 3.2 节），然后提出并对伪随机发生器的定义进行了讨论（见 3.3 节），最后给出了一些简单的构造方法（见 3.4 节）。3.5 节中讨论了更一般化的构造方法。伪随机函数的定义和构造（基于任意伪随机发生器）见 3.6 节。关于伪随机置换的讨论见 3.7 节。

教学提示 混合技术最初用来指出存在于复合抽样之中的计算不可分辨性（见 3.2.3 小节），它在许多有关计算不可分辨性的证明中起着重要的作用。因此，如果你跳过了这个特定的证明，一定要在首次用到混合技术的地方加入对它的讨论。

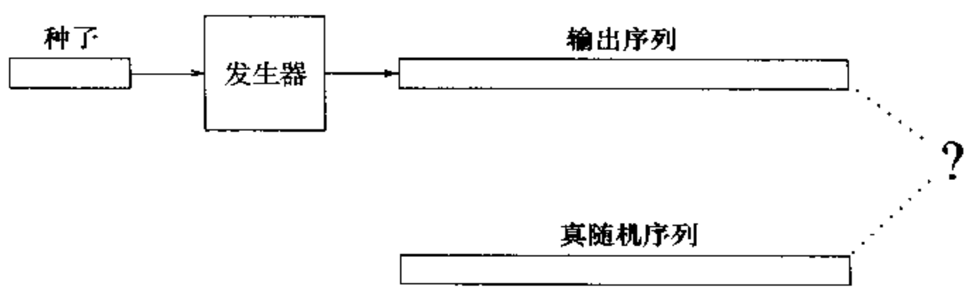


图 3.1 伪随机发生器的一个图解

3.1 启发性讨论

几个世纪以来，随机性问题一直困扰着研究人员。我们相信，计算的概念，特别是有效计算的概念为理解随机性的性质提供了很好的基础。

3.1.1 随机性的计算逼近

早在20世纪60年代初期，Solomonov和Kolmogorov就已提出一种随机性的计算逼近（在20世纪70年代初期又被Chaitin重新提出）。这种近似逼近本质上是“存在的”。不严格地说，如果串 s 的长度（即 $|s|$）等于生成它的最短的程序的长度，我们就认为串 s 是Kolmogorov-随机的。这个最短的程序被认为是串 s 所描述现象的“最简单”的“解释”。因此如果串 s 没有一个“简单”的解释（即一个比 $|s|$ 充分短的解释），我们就认为串 s 是Kolmogorov-随机的。必须指出，目前我们还不能确定一个给定串是否是Kolmogorov-随机的（或更一般地说，Kolmogorov复杂性是一个不可计算的函数）。此外，这种逼近看起来对“伪随机发生器”是没有什么应用的。

本章剩余的部分介绍了另外一种可供选择的随机性的计算逼近。这种近似逼近在20世纪80年代初期被提出。与Kolmogorov逼近相比，新近似逼近在本质上是应用性比较强的。不考虑对一种现象的“解释”，我们考虑这种现象对（外界）环境的影响。简单地说，如果任何有效的观测器都不能将一个串从与它具有相同长度的均匀选择的串中分辨出来，我们就称这个串是伪随机的。下面假设，所有不能被有效算法分辨的对象都认为是等价的，尽管它们本质上是非常不同的（例如，可能有根本不同的（Kolmogorov）复杂性）。此外，新逼近自然地引出了伪随机发生器的概念，这是有许多实际应用的基本概念（特别是在密码学领域）。

3.1.2 伪随机发生器的一个严格逼近

在关于现在真实计算机使用的“伪随机发生器”的讨论中，普遍使用的是启发式近似法，与本书介绍的伪随机发生器的近似逼近正好形成对比。启发式近似法认为“伪随机发生器”是一些程序，这些程序能够产生“可通过”一些特定统计检测的位序列。选择这些程序要通过的统计检测是非常随意的，并且是无任何系统基础的。此外，还可能构造有效的统计检测，使通常使用的“伪随机发生器”不能通过它的验证（特别地，能够分辨具有相同长度的均匀选择串和它们的输出）。因此，在需要“随机”序列的新应用程序使用“伪随机发生器”之前，必须做大量的检测来决定分别用“伪随机发生器”和“真随机源”时应用程序的运行是否相同。对应用程序的任何修改都必须对比地考虑“伪随机发生器”和“真随机源”，因此“伪随机发生器”的非随机性可能从负面影响修改的应用程序（即使它对原程序没有影响）。对于密码编码学中的应用程序，情况就更糟了，因为在这种情况下，只有当攻击者确定后，应用程序才被完全评估。也就是说，在未明确攻击者的情况下不能检测“伪随机发生器”的性能，我们也没有理由假设攻击者使用设计者知道的特定策略。因此，应用这样的“伪随机发生器”来达到密码学的要求，要冒非常

大的危险。

相比之下，这里介绍的伪随机发生器的概念是比较通用的：这种定义下，这些伪随机发生器产生的序列被任何有效的观测器检测，结果都是随机的。从而在任何要求随机串的有效的应用程序中，都可以用伪随机发生器的输出来代替"随机序列"。特别指出，任何高效的攻击者都不能利用伪随机序列来代替"真正的随机序列"。

3.2 计算不可分辨性

如前所述，计算不可分辨性的概念是我们定义伪随机性的基础。因此，我们以对这个基础概念的定义和讨论为起点。

有效计算的概念很自然地引出对象间的一种新的等价关系：如果对象之间不能被任意的有效程序分辨，我们就认为这些对象是计算等价的。认为不可分辨的对象是等价的，这是科学和现实生活共有的基本模式。因此，我们认为引出不可分辨的概念是非常自然的。

3.2.1 定义

计算不可分辨性概念的提出是计算复杂性领域里非常标准的一种方式：它所考虑的对象是无限串序列。因此，如果没有有效的算法能将序列$\{x_n\}_{n\in N}$和$\{y_n\}_{n\in N}$分辨开来，我们就说它们是计算不可分辨的。换句话说，就是没有有效的算法 D 能够接受无限多的 x_n，而拒绝与它们相对应的 y 值(即，对每一个有效算法 D，所有充分大的 n，都有 D 接受 x_n 当且仅当 D 接受 y_n)。这种意义下计算不可分辨的对象在任何实际应用中都可以认为是等价的(因为实际应用都是由有效算法决定的，而后者不能分辨这些对象)。

上述讨论很自然地就推广到了概率背景中。进一步，我们还应该看到这种延伸会产生非常有用的结果。简单地说，如果没有有效算法能够分辨两种分布，我们就说它们是计算不可分辨的。给定有效算法 D，我们考虑 D 接受(例如，对应此输入，输出为 1)服从第一个分布的串的概率。类似地，考虑 D 接受服从第二个分布的串的概率。如果这两个概率非常接近，我们就说 D 不能分辨这两种分布。再者，我们的讨论是关于两个无限分布序列的(而不是两个固定分布)。这样的序列称为一个概率总体。

定义 3.2.1(概率总体)：设 I 是一个可列指标集。我们定义一个**由 I 标记的总体**就是一个由 I 标记的随机变量序列。即，任意 $X=\{X_i\}_{i\in I}$都是一个由 I 标志的总体，这里每一个 X_i 都是一个随机变量。

我们一般用$\mathbb{N}$ 或$\{0,1\}^*$ 的子集作为指标集。特别是在我们的程序中，具有形式 $X=\{X_n\}_{n\in\mathbb{N}}$的总体中的每一个 X_n 都是长度为 poly(n)(n 的多项式)的串，而具有形式 $X=\{X_w\}_{w\in\{0,1\}^*}$的总体中的 X_w 都是长度为 poly($|w|$)的串。本章其余部分将处理以$\mathbb{N}$ 为指标集的总体，而在其他章节(例如，在安全加密和零知识证明的定义中)将处理由串作为指标集的总体。为避免混淆，在这两种情况下，将分别介绍计算不可分辨性的定义中的变量。如果把自然数与它们的一元表示结合起来，这两种情况就可以统一(即，结果$\mathbb{N}$ 和$\{1^n:n\in\mathbb{N}\}$)。

定义 3.2.2(多项式时间不可分辨性)：

1. 由ℕ标记的总体的变量：两个总体 $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathbb{N}}$ 和 $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathbb{N}}$，如果对于每一个概率多项式时间算法 D、每一个正多项式 $p(\cdot)$ 和所有足够大的 n，都有

$$|\Pr[D(X_n,1^n)=1]-\Pr[D(Y_n,1^n)=1]|<\frac{1}{p(n)}$$

则称这两个总体是**在多项式时间内不可分辨的**。

2. 由串集合 S 标记的总体的变量：两个总体 $X \stackrel{\text{def}}{=} \{X_w\}_{w\in S}$ 和 $Y \stackrel{\text{def}}{=} \{Y_w\}_{w\in S}$，如果对于每一个概率多项式时间算法 D、每一个正多项式 $p(\cdot)$ 和所有足够长的 $w\in S$，都有

$$|\Pr[D(X_w,w)=1]-\Pr[D(Y_w,w)=1]|<\frac{1}{p(|w|)}$$

则称这两个总体是在**多项式时间内不可分辨的**。

我们常常将多项式时间不可分辨性简称为计算不可分辨性。

上述定义中的概率是由相应的随机变量 X_i（或 Y_i）以及算法 D 的内部掷币序列（可以是一个概率（不确定）算法）决定的。定义的第二个变量在今后的章节中起着关键的作用，在那里将对它做进一步讨论。在本章其余部分，我们只考虑上述定义中的第一个变量。算法 D 中的辅助输入串 1^n 是为了使得第一个变量与第二个一致。我们指出在特定的情况下，$X_n(Y_n)$ 的长度与 n 是多项式相关的（即，$|X_n|<\text{poly}(n)$ 且 $n<\text{poly}(|X_n|)$），更进一步可以在多项式时间内从其中一个计算出另一个。在这种情况下，给出 1^n 作为辅助输入是多余的。事实上，在整个这一章中，我们都忽略这个辅助输入并假设 n 可以由 X_n 有效地确定。

下面的智力实验（mental experiment）可能是很有意义的。对每一个 $\alpha\in\{0,1\}^*$，考虑输入 α 时算法 D 输出 1 的概率，今后记为 $d(\alpha)$。考虑由两个总体分别决定的 d 的期望；也就是说，设 $d_X(n)=\mathrm{E}[d(X_n)]$，$d_Y(n)=\mathrm{E}[d(Y_n)]$。如果偏差函数 $\delta(n)\stackrel{\text{def}}{=}|d_X(n)-d_Y(n)|$ 对 n 是可忽略的，我们就说 X 和 Y 是 D 不可分辨的。可以回顾一下可忽略的定义，函数 $\mu:\mathbb{N}\to[0,1]$ 称为可忽略的，如果对每一个正多项式 p 和所有足够大的 n 都有 $\mu(n)<1/p(n)$。

几个例子可以帮助我们理解上面的定义。考虑一个算法 D_1，不考虑输入，掷硬币（取值为 0-1）来输出它的结果。显然，对每一个输入，算法 D_1 以恰好 1/2 的概率输出 1，因而不能分辨任何总体对。然后考虑算法 D_2。当且仅当输入串中 0 比 1 多时，输出为 1。因为 D_2 可以在多项式时间内实现，从而如果 X 和 Y 是多项式时间可分辨的，那么偏差 $|\Pr[\text{wt}(X_n)<\frac{n}{2}]-\Pr[\text{wt}(Y_n)<\frac{n}{2}]|$ 对 n 是可忽略的，这里 $\text{wt}(\alpha)$ 表示串 α 中 1 的个数。同样地，关于任意的可以在多项式时间内计算的“串统计量”，多项式时间不可分辨总体必须表现相同的“构形”（以可忽略的错误）。然而，对于多项式时间内不可计算的数量，并不要求多项式时间不可分辨总体都有相同的“构形”（例如，Kolomogorov 复杂性或紧跟在命题 3.2.3 后面的函数）。

3.2.2 统计相关性

计算不可分辨性是概率论中传统概念的一个推广。如果两个总体 $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathbb{N}}$ 和 $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathbb{N}}$ 的统计偏差是可以忽略的，我们就说它们是统计接近的。这里 X 和 Y 之间的统计偏差（或者说是变差距离）如下定义：

$$\Delta(n) \stackrel{\text{def}}{=} \frac{1}{2}\cdot\sum_{\alpha}|\Pr[X_n=\alpha]-\Pr[Y_n=\alpha]| \tag{3.1}$$

显然，如果总体 X 和 Y 是统计接近的，那么它们也是多项式时间不可分辨的（见习题 6）。但反过来是不正确的。特别地：

命题 3.2.3：存在总体 $X=\{X_n\}_{n\in\mathbb{N}}$，使得 X 与均匀总体 $U\stackrel{\text{def}}{=}\{U_n\}_{n\in\mathbb{N}}$ 不是统计接近的，但 X 与 U 是多项式时间不可分辨的。进一步，X_n 可以将它所有的概率质量分配给至多 $2^{n/2}$ 个长为 n 的串。

考虑 U_n 均匀分布在长为 n 的串上。尽管 X 与 U 是多项式时间不可分辨的，我们仍可以定义函数 $f:\{0,1\}^*\rightarrow\{0,1\}$，使得 f 在 X 上平均取 1 而在 U 上平均取 0（例如，$f(x)=1$ 当且仅当 $\Pr[X=x]>0$）。因此，X 与 U 对于函数 f 有不同的“构形”，且在多项式时间内计算 f 是不可能的。

证明：我们申明，对于所有足够大的 n，都存在一个随机变量 X_n，分布在至多 $2^{n/2}$ 个长为 n 的串的集合上，使得对于每一个长为 $2^{n/8}$（即门的数目）的电路 C_n，有

$$|\Pr[C_n(U_n)=1]-\Pr[C_n(X_n)=1]|<2^{-n/8} \tag{3.2}$$

由此申明很容易得出命题的结论，因为多项式时间分辨器（即使是概率多项式时间分辨器；见习题 10（第 1 部分））能够产生至少与分辨偏差一样大的多项式长度电路。

用概率知识证明前面的声明，也就是说，我们实际上指出了特定种类的大多数分布都能够“愚弄”所有长为 $2^{n/8}$ 的电路。特别地，我们指出，如果在 $\{0,1\}^n$ 中均匀地选择包含 $2^{n/2}$ 个串的一个多重集（multi set），方程式 3.2 就会以非常大的概率成立（由多重集的选择决定）。

令 C_n 表示特定的有 n 个输入的电路，$p_n\stackrel{\text{def}}{=}\Pr[C_n(U_n)=1]$。我们在 $\{0,1\}^n$ 中独立均匀地选择 $2^{n/2}$ 个串，记为 $s_1,\ldots,s_{2^{n/2}}$。定义随机变量 ζ_i，使得 $\zeta_i=C_n(s_i)$；也即这些随机变量依赖于相应的 s_i 的选择。利用 Chernoff 界，我们得到

$$\Pr\left[\left|p_n-\frac{1}{2^{n/2}}\cdot\sum_{i=1}^{2^{n/2}}\zeta_i\right|\geq 2^{-n/8}\right]\leq 2e^{-2\cdot 2^{n/2}\cdot(2^{-n/8})^2}<2^{-2^{n/4}} \tag{3.3}$$

因为至多有 $2^{2^{n/4}}$ 个不同的长为 $2^{n/8}$（门的数目）的电路，从而可得出存在序列 $s_1,\ldots,s_{2^{n/2}}\in\{0,1\}^n$，使得对于每一个长为 $2^{n/8}$ 的电路 C，都有

$$\left|\Pr[C_n(U_n)=1]-\frac{\sum_{i=1}^{2^{n/2}}C_n(s_i)}{2^{n/2}}\right|<2^{-n/8}$$

固定这样的序列 s_i，并令 X_n 均匀分布在序列的每个元素上，就可得出结论。■

进一步说明　命题 3.2.3 介绍了计算不可分辨但非统计接近的两个总体。其中一个在多项式时间内是不可构造的（见定义 3.2.5）。有趣的是，只要伪随机发生器存在，既是计算不可分辨的又有能引起注意的统计偏差的一对多项式时间可构造的总体就能够存在。回忆前文，我们发现这个必要条件也是充分条件。（由伪随机发生器能够产生与均匀总体计算不可分辨且非统计接近的在多项式时间内可构造的总体得来）。

初步说明　对上述证明更严密的检查发现，除了可忽略的部分，几乎所有的长为 $2^{n/2}$ 的序列都可以用来定义随机变量 X_n。特别地，方程式 3.3 中的第二个不等式是过分地放大，实际使用了 $2^{-2^{\Omega(n)}}\cdot 2^{-2^{n/4}}$ 的一个上界。观察到大多数序列都没有包含任何副本，我们就可以固定这样的一个序列，因此，X_n 将是服从序列的 $2^{n/2}$ 个截然不同的元素上的均匀分布。

3.2.3　重复实验不可分辨性

由定义 3.2.2，我们知道如果基于单一样本没有有效算法能够将两个总体分辨开来，我们

就称这两个总体是计算不可分辨的。下面指出对于"可有效构造"的总体，基于单一样本的计算不可分辨性就隐含了基于重复样本的计算不可分辨性。首先，我们介绍"重复取样的不可分辨性"和"可有效构造的总体"的定义。

定义 3.2.4(重复取样的不可分辨性)：如果对于每一个概率多项式时间算法 D、任意两个正多项式 $m(\cdot)$ 和 $p(\cdot)$ 以及所有足够大的 n，都有

$$|\Pr[D(X_n^{(1)},\ldots,X_n^{(m(n))})=1]-\Pr[D(Y_n^{(1)},\ldots,Y_n^{(m(n))})=1]|<\frac{1}{p(n)}$$

我们就称总体 $X\stackrel{\text{def}}{=}\{X_n\}_{n\in\mathbb{N}}$ 和 $Y\stackrel{\text{def}}{=}\{Y_n\}_{n\in\mathbb{N}}$ 是**多项式时间取样不可分辨的**。

这里 $X_n^{(1)},\cdots X_n^{(m(n))}$ 是相互独立的并同分布于 X_n，$Y_n^{(1)},\cdots Y_n^{(m(n))}$ 相互独立并同分布于 Y_n，且 $X_n^{(i)}$ 与 $Y_n^{(i)}$ 都是相互独立的。

定义 3.2.5(可有效构造的总体)：如果存在一个概率多项式时间算法 S，使得对于每一个 n，随机变量 $S(1^n)$ 和 X_n 都是同分布的，则称总体 $X\stackrel{\text{def}}{=}\{X_n\}_{n\in\mathbb{N}}$ 是**多项式时间可构造的**。

定理 3.2.6：设 $X\stackrel{\text{def}}{=}\{X_n\}_{n\in\mathbb{N}}$ 和 $Y\stackrel{\text{def}}{=}\{Y_n\}_{n\in\mathbb{N}}$ 是两个多项式时间可构造的总体，并假设 X 和 Y 是多项式时间不可分辨的(见定义 3.2.2)，则 X 和 Y 是多项式时间重复取样不可分辨的(见定义 3.2.4)。

对定理 3.2.6 的另一种叙述如下。对于每一个总体 $Z\stackrel{\text{def}}{=}\{Z_n\}_{n\in\mathbb{N}}$ 和每一个多项式 $m(\cdot)$，定义 Z 和 $m(\cdot)$-乘积为总体 $\{(Z_n^{(1)}),\ldots,Z_n^{m(n)})\}_{n\in\mathbb{N}}$，这里 $Z_n^{(i)}$ 都是来自 Z_n 的相互独立的样本。定理 3.2.6 指出，如果总体 X 和 Y 是多项式时间不可分辨的，且每一个都是多项式时间可构造的，那么对于每一个多项式 $m(\cdot)$，总体 X 的 $m(\cdot)$-乘积和总体 Y 的 $m(\cdot)$-乘积是多项式时间不可分辨的。

上述定理的信息论的类比含义是非常明显的：如果两个总体是统计接近的，那么它们的多项式乘积也是统计接近的(见习题 7)。和常规一样，证明适合计算背景的要求，即可约性理论。可约性理论在本书中首次应用了混合技术。混合技术在说明复合总体的计算不可分辨中起着重要的作用，复合总体的构造建立在更简单的(计算不可分辨性)总体之上。接下来对混合技术的应用学术性要求更高，因此，读者千万不要错过下面的证明。

证明：证明过程用到了可约性理论。我们指出能够利用多个样本分辨总体 X 和 Y 的有效算法存在，就意味着应用一个样本分辨总体 X 和 Y 的有效算法存在。这个蕴涵关系用下面的理论证明，这个理论后面称为"混合理论"。

用反证法，假设存在概率多项式时间算法 D，以及多项式 $m(\cdot)$ 和 $p(\cdot)$，使得对于无限多的 n，都有

$$\Delta(n)\stackrel{\text{def}}{=}|\Pr[D(X_n^{(1)},\ldots X_n^{(m)})=1]-\Pr[D(Y_n^{(1)},\ldots,Y_n^{(m)})=1]|>\frac{1}{p(n)}\quad(3.4)$$

这里 $m\stackrel{\text{def}}{=}m(n)$，并且所有的 $X_n^{(i)}$ 和 $Y_n^{(i)}$ 和定义 3.2.4 中一样。接下来，我们应该提供一个概率多项式时间算法 D'，使得它能够分辨 X 和 Y(在定义 3.2.2 的意义下)，从而推出矛盾。

对每一个 k，满足 $0\leq k\leq m$，我们定义混合随机变量 H_n^k 为一个(长为 m 的)序列，此序列前 k 个分量来自 X_n 的独立随机样本，后 $m-k$ 个来自 Y_n 的独立随机样本。即，

$$H_n^k\stackrel{\text{def}}{=}(X_n^{(1)},\ldots,X_n^{(k)},Y_n^{(k+1)},\ldots,Y_n^{(m)})$$

这里 $X_n^{(1)},X_n^{(k)}$ 是相互独立的并同分布于 X_n，$Y_n^{(k+1)},\ldots Y_n^{(m)}$ 相互独立并同分布于 Y_n，且 $X_n^{(i)}$ 与 $Y_n^{(i)}$ 都是相互独立的。显然，$H_n^m=(X_n^{(1)},\ldots,X_n^{(m)})$，而 $H_n^0=(Y_n^{(1)},\ldots,Y_n^{(m)})$。

通过我们的假设，算法 D 能够分辨两端的混合（即 H_n^0 和 H_n^m）。因为混合的总数是 n 的多项式，两个端点混合之间的不可忽略的偏差（算法 D 的"接受概率"）就转化为一对相邻混合之间的不可忽略的偏差。从而可得出，尽管 D 不是"设计来处理一般混合技术的"，也能够分辨一对相邻的混合。最妙的是，算法 D 能够很容易地修改为算法 D'，从而能够分辨 X 和 Y。细节如下。

以 D 为子程序构造算法 D'。当输入为 α（假设在 X_n 或 Y_n 的取值范围之内）时，算法 D' 如下运行。首先在集合 $\{0,1,...,m-1\}$ 中均匀选择 k，对总体 X 应用有效的取样算法，算法 D' 能够产生 k 个来自 X_n 的独立随机样本，记为 $x^1,...,x^k$。类似地，对总体 Y 应用有效取样算法，算法 D' 产生 $m-k-1$ 个相互独立的来自总体 Y_n 的随机样本，记为 $y^{k+2},...,y^m$。最后算法 D' 调用算法 D，并最后输出 $D(x^1,...,x^k,\alpha,y^{k-2},...,y^m)$。

显然，D' 能在概率多项式时间内实现。证明如下结论也很容易。

结论 3.2.6.1：

$$\Pr[D'(X_n)=1]=\frac{1}{m}\sum_{k=0}^{m-1}\Pr[D(H_n^{k+1})=1]$$

和
$$\Pr[D'(Y_n)=1]=\frac{1}{m}\sum_{k=0}^{m-1}\Pr[D(H_n^k)=1]$$

证明：通过算法 D' 的构造，我们有

$$D'(\alpha)=D(X_n^{(1)},...,X_n^{(k)},\alpha,Y_n^{k+2},...,Y_n^{(m)})$$

这里 k 是从 $\{0,1,...,m-1\}$ 中均匀选择的，利用混合随机变量 H_n^k 的定义，结论得证。

结论 3.2.6.2：对于方程式 3.4 中的 $\Delta(n)$，有

$$|\Pr[D'(X_n)=1]-\Pr[D'(Y_n)=1]|=\frac{\Delta(n)}{m(n)}$$

证明：对第一个等式应用结论 3.2.6.1，我们有

$$\begin{aligned}|\Pr[D'(X_n)=1]-\Pr[D'(Y_n)=1]| &= \frac{1}{m}\cdot\left|\sum_{k=0}^{m-1}\Pr[D(H_n^{k+1})=1]-\Pr[D(H_n^k)=1]\right| \\ &= \frac{1}{m}\cdot|\Pr[D(H_n^m)=1]-\Pr[D(H_n^0)=1]| \\ &= \frac{\Delta(n)}{m}\end{aligned}$$

这里最后一个等式是利用 $H_n^m=(X_n^{(1)},...,X_n^{(m)})$ 和 $H_n^0=(Y_n^{(1)},...,Y_n^{(m)})$，并应用 $\Delta(n)$ 的定义得来的。

由我们的假设，对无限多的 n 有 $\Delta(n)>\frac{1}{p(n)}$，可得出概率多项式时间算法 D' 能够分辨 X 和 Y 与定理的前提相矛盾。从而定理得证。□

混合技术：摘要

考虑一下混合技术（在前述证明中首次使用）是很有意义的。混合技术包括一个特殊类型的"可约性理论"，在这个理论中复合总体的计算不可分辨性是通过基本总体的计算不可分辨性证明的。实际化简是从另一个方向进行的：有效分辨基本总体转化为有效分辨复合总体，变形的基本过程中用到了混合分布。下面的混合随机变量的结构性质在证明中起到了重要的作用：

1. 端点混合随机变量与复合总体相冲突：这个性质是本质的，因为我们想要证明的不可

分辨性是关于复合总体的不可分辨性。

2. 相邻混合与基本总体极易关联：这个性质是必需的，因为我们知道的不可分辨性是基本总体的不可分辨性。我们必须能够将已知的基本总体的计算不可分辨性转化为任意对的相邻混合的计算不可分辨性。特别地，必须能有效地将基本分布取值范围内的串转化为混合分布范围内的串，使得变形将第一个基本分布映射为一个混合变量，而将第二个基本分布映射为相邻的混合变量（在定理 3.2.6 的证明中，假设 X 和 Y 都是多项式时间可构造的实际上是这样一个有效变形的手段）。

3. 混合随机变量的数目很小（即多项式数量）：为了从每对相邻混合的计算不可分辨性推导出端点混合的计算不可分辨性，这个性质是必需的。特别地，可证明的"可分辨性间断"与混合数目是成反比的。

我们指出，在混合证明的过程中，分析并执行了一个有关复合总体的适用于任意混合的著名算法。读者可能很苦恼，这样的算法"不是设计来处理这些混合的"（而仅仅是处理端点混合的）。然而，算法毕竟是算法：一旦存在，我们就可以把它应用到我们选择的任意输入中，并对于任意输入分布分析它的运行。

改进的定理 3.2.6 的进一步注释：定理 3.2.6 的非平凡性的补充说明是，当有一个总体不能有效构造的时候，结论就不成立。实际上，假设两个总体都是可有效构造的，这在定理 3.2.6 的证明中起着非常重要的作用。定理 3.2.6 的信息论模拟指出，对任意两个总体，它们的统计接近就隐含了复合总体的统计接近，请读者比较这两个事实。

3.2.4* 电路族不可分辨性

计算不可分辨性的一个更强的概念是非均匀多项式长度电路族计算不可分辨性的概念，这个概念在后面的章节中将用到。

定义 3.2.7（多项式长度电路族不可分辨性）：

1. 由$\mathbb{N}$标记的总体的变量：两个总体 $X\stackrel{\text{def}}{=}\{X_n\}_{n\in\mathbb{N}}$和 $Y\stackrel{\text{def}}{=}\{Y_n\}_{n\in\mathbb{N}}$，如果对于每一个多项式长度电路族$\{C_n\}_{n\in\mathbb{N}}$、每一个正多项式 $p(\cdot)$和所有足够大的 n 都有

$$|\Pr[C_n(X_n)=1]-\Pr[C_n(Y_n)=1]|<\frac{1}{p(n)}$$

则称这两个总体是**多项式长度电路族不可分辨的**。

2. 串集合 S 标记的总体的变量：两个总体 $X\stackrel{\text{def}}{=}\{X_w\}_{w\in S}$和 $Y\stackrel{\text{def}}{=}\{Y_w\}_{w\in S}$，如果对于每一个多项式长度电路族$\{C_n\}_{n\in\mathbb{N}}$、每一个正多项式 $p(\cdot)$和所有足够长的 $w\in S$，都有

$$|\Pr[C_{|w|}(X_w)=1]-\Pr[C_{|w|}(Y_w)=1]|<\frac{1}{p(|w|)}$$

则称这两个总体是**多项式长度电路族不可分辨的**。

我们说由串集合 S 标记的总体的变量与下面（看似更强）的条件等价：

对每一个多项式 $s(\cdot)$，每一个电路族集合$\{C_w\}_{w\in S}$，满足 C_w 的长度至多为 $s(|w|)$，每一个正多项式 $p(\cdot)$和所有足够长的 w，有

$$|\Pr[C_w(X_w)=1]-\Pr[C_w(Y_w)=1]|<\frac{1}{p(|w|)} \tag{3.5}$$

我们指出第二个要求不比定义中的要求更强：也就是说，如果第二个要求不满足，则第一

个也不满足。假设对一些多项式 s 和 p，存在无限多的 w 不满足方程式 3.5。那么就存在无限集 N，使得对每一个 $n\in N$，存在一个串 $w_n\in\{0,1\}^n$ 不满足方程式 3.5。令 $C'_n \stackrel{\text{def}}{=} C_{w_n}$，我们得到一个与定义的要求相矛盾的结论。

我们注意到，倘若前述定义中的概率电路族不能增加它的幂次（见习题 8），那么依据我们的元定理（见 1.3.3 小节），通过多项式长度电路族的不可分辨性（见定义 3.2.7 中的两个定义）就隐含了通过概率多项式时间机的不可分辨性（见定义 3.2.2 中的两个定义）；见习题 10。反过来却不对（见习题 10）。最后，我们指出多项式长度电路族的不可分辨性在重复试验下也成立，即使两个总体都不可有效构造（见习题 9）。

3.2.5　伪随机总体

总体对计算不可分辨的一个特殊而重要的情况是其中一个总体是均匀的。与均匀总体计算不可分辨的总体被认为是伪随机的。记 U_m 表示服从长为 m 的串集合上均匀分布的随机变量，则总体 $\{U_n\}_{n\in\mathbb{N}}$ 称为标准均匀总体。再者，我们称具有形式 $\{U_{l(n)}\}_{n\in\mathbb{N}}$ 的总体也是均匀的，这样命名比较方便，这里 $l:\mathbb{N}\to\mathbb{N}$。

定义 3.2.8（伪随机总体）：如果存在一个均匀总体 $U=\{U_{l(n)}\}_{n\in\mathbb{N}}$，使得 X 与 U 在多项式时间内是不可分辨的，则称总体 $X=\{X_n\}_{n\in\mathbb{N}}$ 是**伪随机**的。

我们强调这里的 $|X_n|$ 不必等于 n（而 $|U_m|=m$）。实际上，对于定义 3.2.8 中的多项式时间可计算的 $l:\mathbb{N}\to\mathbb{N}$ 和 $X=\{X_n\}_{n\in\mathbb{N}}$，$|X_n|$ 以很高的概率等于 $l(n)$。

在前面的定义中，以及本书剩余的部分中，伪随机性实际上是关于多项式时间的伪随机性。

3.3　伪随机序列发生器定义

这里定义的伪随机总体在有效的应用程序中可以代替均匀总体，且性能的降低是可忽略不计的（否则有效的应用程序就变成了能把伪随机总体从均匀总体中区分出来的有效的识别器）。这样的替换是有用的仅当生成伪随机总体的费用低于生成对应的均匀总体的费用。产生一个总体的费用包含几个方面。标准费用包括时间和空间复杂度。然而，在随机化算法的上下文中，尤其是在生成概率总体的上下文中，主要的费用在于所使用算法的随机资源的数量和质量。特别地，在许多应用中（特别是密码学里）期望利用尽可能少的真正的随机性来产生伪随机总体。这就引发了伪随机序列发生器的定义。

3.3.1　伪随机发生器的标准定义

定义 3.3.1（伪随机发生器，标准定义）：**伪随机发生器**即为满足以下两个条件的确定性多项式时间算法 G。

1. 扩展性：存在一个函数 $l:\mathbb{N}\to\mathbb{N}$，使得对于所有 $n\in\mathbb{N}$ 有 $l(n)>n$，对于所有 $s\in\{0,1\}^*$ 有 $|G(s)|=(|s|)$。

2. 伪随机性：总体$\{G(U_n)\}_{n\in\mathbb{N}}$是伪随机的。

函数 l 被称为 G 的扩展因子。

发生器的输入 s 是它的种子。扩展条件要求算法 G 把 n 比特长的种子映射成 $l(n)$ 比特长的字符串，其中 $l(n)>n$。伪随机性条件则要求 G 作用于均匀选择的种子而得到的输出分布与均匀分布在多项式时间内是不可分辨的，尽管输出分布与均匀分布不是统计接近的。特别地，根据习题 5(第一个等式)，我们可以得到 $G(U_n)$ 和 $U_{l(n)}$ 的统计差异的界限如下：

$$\frac{1}{2}\sum_x |\Pr[U_{l(n)}=x]-\Pr[G(U_n)=x]| = \max_S\{\Pr[U_{l(n)}\in S]-\Pr[G(U_n)\in S]\}$$
$$\geq \Pr[U_{l(n)}\notin\{G(s):s\in\{0,1\}^n\}]$$
$$\geq (2^{l(n)}-2^n)\cdot 2^{-l(n)}$$
$$= 1-2^{-(l(n)-n)}\geq\frac{1}{2}$$

其中最后一个不等式利用了 $l(n)\geq n+1$。请注意，若 $l(n)\geq 2n$，则统计差异至少是 $1-2^{-n}$。

关于扩展因子 $l:\mathbb{N}\to\mathbb{N}$，上述定义是可以接受的。只需断言 $l(n)\geq n+1$ 和 $l(n)\leq \text{poly}(n)$(也可以证明 $l(n)$ 在 n 的多项式时间内是可计算的；例如通过计算 $|G(1^n)|$)。很明显，扩展因子为 $l(n)=n+1$ 的伪随机发生器在实际运用中几乎没什么作用，因为它在硬币投掷中没有提供重要的补救。幸运的是，正如下一小节将要说明的那样，扩展因子比较小的平滑伪随机发生器可以用来构造具有多项式扩展因子的伪随机发生器。因此，对每两个可以在 poly(n)时间内计算的扩展因子 $l_1:\mathbb{N}\to\mathbb{N}$ 和 $l_2:\mathbb{N}\to\mathbb{N}$ 来说，扩展因子为 l_1 的伪随机发生器存在当且仅当扩展因子为 l_2 的伪随机发生器存在。这个结论可以按如下方式证明：对每个多项式 $p(\cdot)$，可以利用任一扩展因子为 $l_1(n)\stackrel{\text{def}}{=}n+1$ 的伪随机发生器构造一个扩展因子为 $p(n)$ 的伪随机发生器。需要注意的是，扩展因子为 $l_1(n)\stackrel{\text{def}}{=}n+1$ 的伪随机发生器可能源于任意伪随机发生器。

如前所述的每个伪随机发生器都有一个规定的扩展函数。在 3.3.3 小节中，我们将考虑"不定长输出伪随机发生器"，对给定的随机种子，它可以产生一个无限比特序列，使得每个多项式长的前缀都是伪随机的。

3.3.2 增加扩展因子

给定一个扩展因子为 $l_1(n)=n+1$ 的伪随机发生器 G_1，我们可以按如下方式构造一个扩展因子为任意多项式的伪随机发生器 G。

构造法 3.3.2：设 G_1 是一个确定性的多项式时间算法，它能把长为 n 的字符串映射到长为 $n+1$ 的字符串，设 $p(\cdot)$是一个多项式。定义 $G(s)=\sigma_1\cdots\sigma_{p(|s|)}$，其中 $s_0\stackrel{\text{def}}{=}s$，$\sigma_i$ 是 $G_1(s_{i-1})$ 的第一个比特，且 s_i 是 $G_1(s_{i-1})$ 的长为 $|s|$ 比特的前缀，$1\leq i\leq p(|s|)$。也就是说，当输入 s 时，算法 G 按如下方式进行：

> 设 $s_0=s$，$n=|s|$。
> 对于 $i=1$ 到 $p(n)$，
> $\sigma_i s_i \leftarrow G_1(s_{i-1})$，其中 $\sigma_i\in\{0,1\}$，$|s_i|=|s_{i-1}|$。
> $\sigma_1\sigma_2\cdots\sigma_{p(|s|)}$。

构造过程如图 3.2 所示：当输入 s 时，算法 G 使用算法 G_1 $p(|s|)$次，每次输入一个新的种子。G_1 作用于当前种子能产生一个新的种子（用于下一个循环）和一个附加位（将立即输出）。第一个循环的种子是 s 本身。第 i 个循环的种子是从上一循环的 G_1 得到的字符串的长为 $|s|$ 的前缀。显然，G 是多项式时间可计算的，并且能把长为 n 的输入扩展为长为 $p(n)$ 的输出字符串。

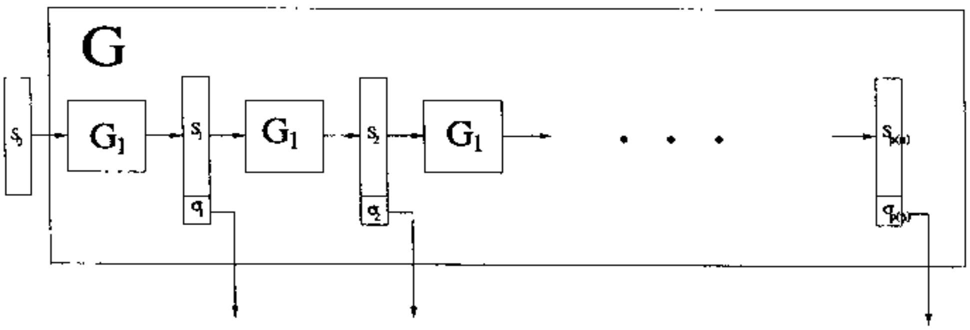


图 3.2 输入种子 $s_0 \in \{0,1\}^n$ 时的构造 3.3.2

定理 3.3.3：设 G_1，$p(\cdot)$和 G 如构造 3.3.2 中所述，且满足 $p(n)>n$。如果 G_1 是伪随机发生器，那么 G 也是。

直观地，通过利用一个随机过程（当输入一个均匀分布的长为 n 的字符串时，此过程输出一个均匀分布的长为$(n+1)$的字符串）替换每个 G_1 的应用程序，可以证明 G 的伪随机性和G_1 的伪随机性一致。简单地说，此随机过程的单个应用程序和 G_1 的单个应用程序是不可分辨的意味着此随机过程的多项式个应用程序和 G_1 的多项式个应用程序是不可分辨的。这一结论可以利用混合技术来证明。

证明：使用反证法。假设 G 不是伪随机发生器，从而可得总体$\{G(U_n)\}_{n\in\mathbb{N}}$和$\{U_{p(n)}\}_{n\in\mathbb{N}}$不是多项式时间内不可分辨的。接着我们将证明总体$\{G_1(U_n)\}_{n\in\mathbb{N}}$和$\{U_{n+1}\}_{n\in\mathbb{N}}$也是多项式时间内不可分辨的，这就与 G_1 是一个扩展因子为 $l_1(n)=n+1$ 的伪随机发生器前提相矛盾。此结论可以通过混合技术来证明。

对每个 $0\le k\le p(n)$的 k，我们定义一个混合 H_n^k 为均匀选择的 k 比特长的字符串和 $G(U_n)$的$(p(n)-k)$比特长的前缀的串联。记字符串 α 的 j 比特长的前缀为 $pref_j(\alpha)$，其中 $j\le|\alpha|$；并记字符串 x 和 y 的串联为 $x\cdot y$，有

$$H_n^k \stackrel{\text{def}}{=} U_k^{(1)} \cdot \text{pref}_{p(n)-k}(G(U^{(2)})) \tag{3.6}$$

其中 $U_k^{(1)}$ 和 $U_n^{(2)}$ 是独立随机变量（其中第一个变量在$\{0,1\}^k$ 上均匀分布，第二个变量在$\{0,1\}^n$ 上均匀分布）。

图 3.3 描述了另一种看待混合 H_n^k 的方法：从构造 3.3.2 开始，选取一个在$\{0,1\}^n$ 上均匀分布的 s_k，在$\{0,1\}^k$ 上均匀分布的 $\sigma_1 \ldots \sigma_k$，对 $i=k+1,\ldots,p(n)$可以像构造中那样得到 $\sigma_i s_i=G_1(s_{i-1})$。

在这个问题上，H_n^0 等于 $G(U_n)$，而 $H_n^{p(n)}$ 等于 $U_{p(n)}$是很明显的。可以证明，如果算法 D 可以分辨端点混合，那么 D 也能分辨两个相邻的混合（由于混合的总数是 n 的多项式，两个端点混合之间的非不可忽略的差异转变为某两个相邻混合之间的非不可忽略的差异）。最关键的是，利用相邻混合的构造，很容易修正算法 D，使之可以分辨总体$\{G_1(U_n)\}_{n\in\mathbb{N}}$和$\{U_{n+1}\}_{n\in\mathbb{N}}$。

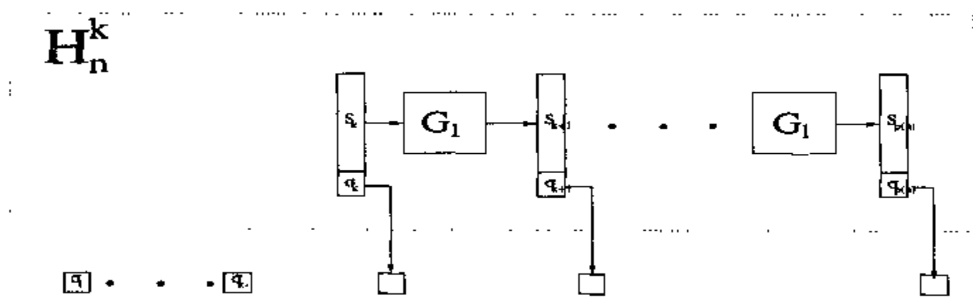


图 3.3 修改构造 3.3.2 得到的混合 H_n^k

此证明方法的核心是相邻混合的可分辨性与 $G_1(U_n)$ 和 U_{n+1} 的可分辨性有关。如前文所述，这个关系源于相邻混合的构造。下面让我们深入理解一下混合 H_n^k 和 H_n^{k-1}，其中 $0\leq k\leq p(n)-1$。此概念的另一方面也是有用的：设字符串 α 的 j 比特长的后缀为 $suff_j(\alpha)$，其中 $j\leq|\alpha|$。首先证明对每个 $x\in\{0,1\}^n$，有

$$pref_{j+1}(G(x)) = pref_1(G_1(x)) \cdot \mathrm{pref}_j(G(\mathrm{suff}_n(G_1(x)))) \tag{3.7}$$

于是（进一步证明如下），

$$H_n^k = U_k^{(1)} \cdot \mathrm{pref}_{(p(n)-k-1)+1}(G(U_n^{(2)}))$$
$$\equiv U_k^{(1)} \cdot \mathrm{pref}_1(G_1(U_n^{(2)})) \cdot \mathrm{pref}_{p(n)-k-1}(G(\mathrm{suff}_n(G_1(U_n^{(2)}))))$$
$$H_n^{k+1} = U_{k+1}^{(1)} \cdot \mathrm{pref}_{p(n)-(k+1)}(G(U_n^{(2)}))$$
$$\equiv U_k^{(1)} \cdot \mathrm{pref}_1(U_{n+1}^{(3)}) \cdot \mathrm{pref}_{p(n)-k-1}(G(\mathrm{suff}_n(U_{n+1}^{(3)})))$$

因此，区分 H_n^k 和 H_n^{k+1} 的能力就转变成了区分 $G_1(U_n^{(2)})$ 和 $U_{n+1}^{(3)}$ 的能力，我们均匀地选取 $r\in\{0,1\}^k$，并把"混合识别器"作用于 $r\cdot\mathrm{pref}_1(\alpha)\cdot\mathrm{pref}_{p(n)-k-1}(G(\mathrm{suff}_n(\alpha)))$。详细证明如下。

首先我们将重申并进一步证明上述等式。我们从记录上文提及的算子这一概念入手。对每个 $k\in\{0,1,\ldots,p(n)-1\}$ 和 $\alpha\in\{0,1\}^{n+1}$，令

$$f_{p(n)-k}(\alpha) \stackrel{\text{def}}{=} \mathrm{pref}_1(\alpha)\cdot\mathrm{pref}_{p(n)-k-1}(G(\mathrm{suff}_n(\alpha))) \in \{0,1\}^{p(n)-k} \tag{3.8}$$

结论 3.3.3.1（$G_1(U_n)$ 和 U_{n+1} 分别与 H_n^k 和 H_n^{k+1}）：相对应

1. H_n^k 的分布等价于 $U_k^{(1)}\cdot f_{p(n)-k}(G_1(U_k^{(2)}))$。

2. H_n^{k-1} 的分布等价于 $U_k^{(1)}\cdot f_{p(n)-k}(U_{n+1}^{(3)})$。

证明：对任意 $x\in\{0,1\}^n$，令 $\sigma=\mathrm{pref}_1(G_1(x))$，$y=\mathrm{suff}_n(G_1(x))$（即 $\sigma y=G_1(x)$）。那么，通过 G 的构造可得 $G(x)=\sigma\cdot\mathrm{pref}_{p(n)-1}(G(y))$。这就证明了等式 3.7；也就是说，对每个 $j\geq0$ 有 $\mathrm{pref}_{j+1}(G(x))=\mathrm{pref}_1(G_1(x))\cdot\mathrm{pref}_j(G(\mathrm{suff}_n(G_1(x))))$。现在我们介绍此结论的两个部分：

1. 结合 H_n^k 的定义和等式 3.7，有

$$H_n^k = U_k^{(1)} \cdot \mathrm{pref}_{(p(n)-k-1)+1}(G(U_n^{(2)}))$$
$$= U_k^{(1)} \cdot \mathrm{pref}_1(G_1(U_n^{(2)})) \cdot \mathrm{pref}_{p(n)-k-1}(G(\mathrm{suff}_n(G_1(U_n^{(2)}))))$$
$$= U_k^{(1)} \cdot f_{p(n)-k}(G_1(U_n^{(2)}))$$

这就构成了论题的第一部分。

2. 结合 H_n^{k-1} 的定义和等式 3.7，有

$$H_n^{k+1} = U_{k+1}^{(1)} \cdot \mathrm{pref}_{p(n)-(k+1)}(G(U_n^{(2)}))$$

$$= U_k^{(1')} \cdot U_1^{(1'')} \cdot \mathsf{pref}_{p(n)-k-1}(G(\mathsf{suff}_n(U_{n+1}^{(2')})))$$
$$= U_k^{(1')} \cdot \mathsf{pref}_1(U_{n+1}^{(2')}) \cdot \mathsf{pref}_{p(n)-k-1}(G(\mathsf{suff}_n(U_{n+1}^{(2')})))$$
$$= U_k^{(1')} \cdot f_{p(n)-k}(U_{n+1}^{(2')})$$

从而得到两部分。

因此，如果把 $f_{p(n)-k}$ 作用于输入，再用一个长为 k 的均匀选择的字符串在前面扩充输出，最后把混合识别器作用于得到的字符串，这样区分 $G_1(U_n)$ 和 U_{n-1} 的问题就简化为区分相邻混合（即 H_n^k 和 H_n^{k+1}）的问题。细节如下。

使用反证法，假设 G 不是伪随机发生器。假设 D 是一个概率多项式算法，对某一多项式 $q(\cdot)$ 和无穷多个 n，有

$$\Delta(n) \stackrel{\text{def}}{=} |\mathsf{Pr}[D(G(U_n) = 1 - \mathsf{Pr}[D(U_{p(n)}) = 1]| > \frac{1}{q(n)}$$

我们通过构造一个可以区分 $G_1(U_n)$ 和 U_{n+1} 的概率多项式时间算法 D' 得到一个矛盾。

算法 D' 把 D 作为子程序。当输入 $\alpha \in \{0,1\}^{n+1}$ 时，算法 D' 按如下方式进行：首先，D' 选择一个在集合 $\{0,1,\ldots,p(n)-1\}$ 上均匀分布的整数 k，接着在 $\{0,1\}^k$ 上均匀选择一个 β，最后输出 $D(\beta \cdot f_{p(n)-k}(\alpha))$ 而终止，其中 $f_{p(n)-k}$ 和等式 3.8 中定义的一样。

显然，D' 可以在概率多项式时间内实现（特别地，$f_{p(n)-k}$ 可以通过结合计算 G 的算法和平凡的字符串操作来实现）。剩下工作的就是分析输入各种分布的 $G_1(U_n)$ 和 U_{n-1} 时 D' 的性能。

论题 3.3.3.2：

$$\mathsf{Pr}[D'(G_1(U_n)) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \mathsf{Pr}[D(H_n^k) = 1]$$

和

$$\mathsf{Pr}[D'(U_{n+1}) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \mathsf{Pr}[D(H_n^{k+1}) = 1]$$

证明：通过构建 D'，我们可得，对每个 $\alpha \in \{0,1\}^{n+1}$，有

$$\mathsf{Pr}[D'(\alpha) = 1] = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \mathsf{Pr}[D(U_k \cdot f_{p(n)-k}(\alpha)) = 1]$$

根据论题 3.3.3.1，结论得证。

设 $d^k(n)$ 为输入混合 H_n^k 时 D 输出 1 的概率（即 $d^k(n) \stackrel{\text{def}}{=} \mathsf{Pr}[D(H_n^k)=1]$）。回顾上面的论题：$H_n^0$ 等于 $G(U_n)$，而 $H_n^{p(n)}$ 等于 $U_{p(n)}$。因此，$d^0(n) = \mathsf{Pr}[D(G(U_n))=1]$，$d^{p(n)}(n) = \mathsf{Pr}[D(U_{p(n)})=1]$，且 $\Delta(n) = |d^0(n) - d^{p(n)}(n)|$。再结合论题 3.3.3.2 可得

$$\begin{aligned}
&|\mathsf{Pr}[D'(G_1(U_n)) = 1] - \mathsf{Pr}[D'(U_{n+1}) = 1]| \\
&= \frac{1}{p(n)} \cdot \left| \left(\sum_{k=0}^{p(n)-1} d^k(n)\right) - \left(\sum_{k=0}^{p(n)-1} d^{k+1}(n)\right) \right| \\
&= \frac{|d^0(n) - d^{p(n)}(n)|}{p(n)} \\
&= \frac{\Delta(n)}{p(n)}
\end{aligned}$$

根据我们（相反的）假设，对无穷多个 n 有 $\Delta(n) > \frac{1}{q(n)}$，从而推出与 G_1 的伪随机性矛盾。

3.3.3* 不定长输出的伪随机发生器

前面定义的伪随机发生器(即在定义 3.3.1 中定义的)预先规定了扩展的数目。也就是说,一旦发生器和种子固定了,伪随机序列的长度也随之固定。下面将给出一个更复杂的定义,它可以产生"灵活的"伪随机序列;也就是说,对任意固定的种子,可以定义一个无限序列使之满足以下两个条件。

1. 可以在多项式时间内产生此序列的任意前缀,该多项式是这个种子和这一前缀的长度的多项式。

2. 对一个均匀选择的 n 比特长的种子,相应输出序列的任意 poly(n)比特的前缀是伪随机的。

换而言之:

定义 3.3.4(不定长输出的伪随机发生器):满足以下两个条件的确定性多项式时间算法 G 是**不定长输出的伪随机发生器**。

1. 不定长输出:对所有 $s\in\{0,1\}^*$ 和 $t\in\mathbb{N}$,$|G(s,1^t)|=t$ 成立,且 $G(s,1^t)$ 是 $G(s,1^{t+1})$ 的前缀。

2. 伪随机性:对每个多项式 p 来说,总体 $\{G(U_n,1^{p(n)})\}_{n\in\mathbb{N}}$ 是伪随机的。

对构造 3.3.2 稍作修改可得如下定理。

定理 3.3.5:如果伪随机发生器存在,那么不定长输出的伪随机发生器亦存在。

按类似方法,可以修正 3.4 节提出的所有构造,以获得不定长输出的伪随机发生器。事实上,对所有构造我们都可以保持一个隐蔽状态,此状态允许在种子的长度的多项式时间内产生序列的下一个比特,而不必考虑当前已经产生的比特数。这就引发了一个如习题 21 定义和研究的即时发生器的概念。

3.3.4 伪随机发生器的适用性

伪随机发生器的显著性质就是它是一个有效的"随机性的扩大器/增强器"。利用很少的随机性(表现为随机选择种子),它们就能产生很长的对任一有效的观察者来说看似随机的序列。因此伪随机发生器的输出可以在任何需要这样(即"随机的")序列的应用程序中代替"真正随机的序列",这就是这样的应用程序被视为一个识别器的原因。换句话说,如果某一有效算法用伪随机序列代替随机序列时会导致不可忽略的性能下降,那么这一算法很容易被修改为一个识别器,此识别器就可以说明后一序列不是伪随机的。

伪随机发生器概念的通用性在实践中有重要作用。一旦我们确定算法是伪随机发生器,我们就能在每个需要"随机序列"的有效应用程序中使用它,而不必在专门的新应用程序中测试此发生器的性能。

伪随机发生器在密码学中的作用是不可估量的(最重要的部分将在以后的章节中给出)。伪随机发生器在密码学中如此重要的原因在于,所有密码学任务的实现都需要许多"高质量的随机性"。因此,以低代价产生、交换和共享大量的"高质量的随机比特"的过程是首要问题。伪随机发生器允许我们以仅产生(或交换/共享)n 个随机比特的代价产生(或交换/共享)poly

(*n*)个伪随机比特!

3.3.5 伪随机性和不可预测性

伪随机序列的主要性质是不可预测性,这一性质使得在一些密码学应用中该序列的使用合理化。简而言之,如果给定了某一序列的前缀,没有任何有效的算法能以大于$\frac{1}{2}$的非不可忽略的成功概率猜测出它的下一个比特,那么这个序列就是不可预测的。也即:

定义 3.3.6(不可预测性):如果对每个概率多项式时间算法 A、每个正多项式 $p(\cdot)$ 和所有足够大的 n,有

$$\Pr[A(1^{|X_n|}, X_n) = \text{next}_A(X_n)] < \frac{1}{2} + \frac{1}{p(n)}$$

其中当输入$(1^{|x|}, x)$时算法 A 只读 x 中 $i<|x|$ 的比特,那么 $\text{next}_A(x)$ 就返回 x 的 $i+1$ 比特;反之返回一个均匀选择的比特(即防止 A 读整个字符串 x)。

和 x 一起输入 $1^{|x|}$ 的作用是为了允许算法在读 x 前预测 x 的长度(且能在那个长度的多项式时间内完成)。为了防止 A 读所有的 x,它必须能够猜出一个完全随机的比特,且猜测成功的概率一定不能大于$\frac{1}{2}$(或者可以禁止 A 读取所有输入;参见习题 20)。当然我们感兴趣的情况还是 A 不读取整个输入而是根据 x 的前 i 比特猜测第 $i+1$ 比特。如果没有任何概率多项式时间算法能以高于$\frac{1}{2}$的非不可忽略的概率完成后一任务,那么这个总体就被称为在多项式时间内是不可预测的。

直观地,伪随机总体在多项式时间内是不可预测的(因为所有均匀总体都是如此)。反之也成立。也即,只有伪随机总体在多项式时间内是不可预测的。

定理 3.3.7(伪随机性对比于不可预测性):总体$\{X_n\}_{n\in\mathbb{N}}$是伪随机的当且仅当它在多项式时间内不可预测。

必要性证明:要证明伪随机性隐含了不可预测性,实际上就是要证明前面提及的直观知识。由于总体 $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathbb{N}}$是伪随机的,所以它在多项式时间内和某一均匀总体是不可分辨的。显然,均匀总体在多项式时间内是不可预测的;事实上,不管预测算法的时间约束条件是多少,它都是不可预测的。于是,总体 X 必须也是在多项式时间内不可预测的,否则我们就能在多项式时间内把总体 X 和均匀总体区别开来(这与前提矛盾)。详细证明如下。

为了简化(而又不失一般性),可以假设总体 $X=\{X_n\}_{n\in\mathbb{N}}$满足$|X_n|=n$,于是它在多项式时间内同标准的均匀总体$\{X_n\}_{n\in\mathbb{N}}$是不可分辨的。相反地,假设$\{X_n\}_{n\in\mathbb{N}}$可以通过算法 A 在多项式时间内预测;也就是说,对某一多项式 p 和无穷多个 n,有

$$\Pr[A(1^n, X_n) = \text{next}_A(X_n)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

于是 A 很容易被转变成一个识别器,记为 D,它按如下方式运行:输入 y,识别器在输入为$(1^{|y|}, y)$的条件下调用 A,并记下 A 实际读到的比特数及 A 对下一比特的预测。如果预测是正确的,D 就输出 1,反之输出 0。显然有,

$$\Pr[D(X_n) = 1] = \Pr[A(1^n, X_n) = \text{next}_A(X_n)]$$

$$\geq \frac{1}{2} + \frac{1}{p(n)}$$

然而

$$\Pr[D(U_n) = 1] = \Pr[A(1^n, U_n) = \text{next}_A(U_n)]$$
$$\leq \frac{1}{2}$$

因此，$\Pr[D(X_n)=1] - \Pr[D(U_n)=1] \geq 1/p(n)$，这与前提$\{X_n\}_{n\in \mathbb{N}}$是伪随机的相矛盾。必要性得证。■

充分性证明：证明充分性（即不可预测性包含伪随机性）要复杂些。事实上，这种情况下直观知识更不明显。由信息论对等体提供的一个动机是：无法被预测（抛开可计算性问题时）的0-1随机变量的最佳序列是随机变量独立且在$\{0,1\}$上均匀分布的序列。在当前状况下，可计算对等体也成立，但是其证明也更加复杂。此证明过程要结合混合技术的使用和要证明的结论的特殊情况。简而言之，特殊情况是指两个总体$Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in \mathbb{N}}$和$Y' \stackrel{\text{def}}{=} \{Y'_n\}_{n\in \mathbb{N}}$，其中$Y'_n$是忽略$Y_n$的最后一比特而得到的。结论就是，如果$Y'$是伪随机的，且$Y$在多项式时间内是不可预测的，那么$Y$是伪随机的。根据此结论，如果$X_n$的$i$比特长的前缀是伪随机的，且$X_n$的$(i+1)$比特长的前缀是多项式时间内不可预测的，那么后者也是伪随机的。下面我们把这个直观的知识变为严格的证明。

利用反证法，假设$X=\{X_n\}_{n\in \mathbb{N}}$不是伪随机的。为了简化（且不失一般性），我们再次假设$|X_n|=n$。于是存在一个概率多项式时间算法D，它能把X同标准均匀总体$\{U_n\}_{n\in \mathbb{N}}$区分开来；也就是说，对某个多项式$p$和无穷多个$n$，有

$$|\Pr[D(X_n) = 1] - \Pr[D(U_n) = 1]| \geq \frac{1}{p(n)} \tag{3.9}$$

不失一般性，假设对无穷多个n，有

$$\Pr[D(X_n) = 1] - \Pr[D(U_n) = 1] \geq \frac{1}{p(n)} \tag{3.10}$$

证明绝对值的下降性：假设S是满足3.9式的n的无限集，则S一定包含一个n的无限子集，对此集合的n来说，$|\Pr[D(X_n)=1] - \Pr[D(U_n)=1]|$或是正的或是负的。不失一般性，我们可以假设它是正的。否则我们可以通过反转其输出来修改D。

对每个满足3.10式的n，我们定义$n+1$个混合。第i个混合$(i=0,1,\ldots,n)$记为H_n^i，构成了X_n的i比特长的前缀，其后紧跟着U_n的$(n-i)$比特长的后缀。上文所述的前提假设意味着存在一对在多项式时间内不可分辨的相邻混合。实际上对一对“随机”的相邻混合来说，此结论在大多数情况下是成立的。

结论 3.3.7.1：对每个满足3.10式的n，有

$$\frac{1}{n} \cdot \sum_{i=0}^{n-1} (\Pr[D(H_n^{i+1}) = 1] - \Pr[D(H_n^i) = 1]) \geq \frac{1}{p(n) \cdot n}$$

证明：可直接通过3.10式和混合的定义来证明。特别地，我们利用了$H_n^n = X_n$和$H_n^0 \equiv U_n$这一事实。□

结论 3.3.7.1给出了一个预测$\{X_n\}_{n\in \mathbb{N}}$的下一比特的简单算法，记为$A$。此算法在$\{0,1,\ldots,n-1\}$上均匀地选择一个$i$，从$X_n$中读取$i$比特，再把这$i$比特和$n-i$个均匀选取的比特相连，得到$n$比特字符串，然后输入这$n$比特的字符串来调用$D$。如果$D$的响应为1，那么$A$

的预测值设为这 $n-i$ 个随机比特的第一位的值；反之，设为互补值。其原因如下：如果这 $n-i$ 个随机比特的第一位恰好等于 X_n 的第 $n+1$ 比特，那么 A 在其输入的分布状态等价于 H_n^{i+1} 时被调用；反之，如果这 $n-i$ 个随机比特的第一位恰好等于（X_n 的第 $n+1$ 比特的）互补值，那么 A 在其输入的分布状态等价于 Z 时被调用，其中 Z 和 H_n^{i+1} 的区别要比 Z 和 H_n^i 明显得多（即 H_n^i 等于 Z 的概率为 $\frac{1}{2}$，H_n^{i+1} 反之）。细节如下。

首先我们给出算法 A 更精确的描述。输入 1^n 和 $x=x_1\cdots x_n$，算法 A 按如下步骤进行：

1. 在 $\{0,1,\ldots,n-1\}$ 上均匀地选择一个 i；
2. 选择在 $\{0,1\}$ 上独立且均匀分布的 $r_{i+1},\ldots,r_n$；
3. 如果 $D(x_1\cdots x_i r_{i+1}\cdots r_n)=1$，就输出 r_{i+1}，反之输出 $1-r_{i+1}$。

结论 3.3.7.2：对每个满足 3.10 式的 n，有

$$\Pr[A(1^n,X_n)=\text{next}_A(X_n)]>\frac{1}{2}+\frac{1}{p(n)\cdot n}$$

证明：X_n 的第 j 比特记为 X^j，在 $\{0,1\}$ 上均匀分布的 $n-i$ 个独立随机变量记为 $R^{i+1},\ldots,R^n$。由 A 的定义和 $\Pr[X^{i+1}=R^{i+1}]=\frac{1}{2}$，可得

$$\begin{aligned}
s_A(n)&\stackrel{\text{def}}{=}\Pr[A(1^n,X_n)=\text{next}_A(X_n)]\\
&=\frac{1}{n}\cdot\sum_{i=0}^{n-1}(\Pr[D(X^1\cdots X^iR^{i+1}\cdots R^n)=1\,\&\,R^{i+1}=X^{i+1}]\\
&\quad+\Pr[D(X^1\cdots X^iR^{i+1}\cdots R^n)=0\,\&\,1-R^{i+1}=X^{i+1}])\\
&=\frac{1}{2n}\sum_{i=0}^{n-1}(\Pr[D(X^1\cdots X^iX^{i+1}R^{i+2}\cdots R^n)=1]\\
&\quad+1-\Pr[D(X^1\cdots X^i\overline{X}^{i+1}R^{i+2}\cdots R^n)=1])
\end{aligned}$$

其中 $\overline{X}^{i+1}\stackrel{\text{def}}{=}1-X^{i+1}$。根据如下事实：$H_n^i$ 的分布等价于以 $\frac{1}{2}$ 的概率取 $H_n^{i+1}=X^1\cdots X^iX^{i+1}R^{i+2}\cdots R^n$，而余下的取 $Z\stackrel{\text{def}}{=}X^1\cdots X^i\overline{X}^{i+1}R^{i+2}\cdots R^n$ 时得到的分布函数，我们可以得到：

$$\Pr[D(H_n^i)=1]=\frac{\Pr[D(H_n^{i+1})=1]+\Pr[D(Z)=1]}{2}$$

这就意味着 $\Pr[D(Z)=1]=2\Pr[D(H_n^i)=1]-\Pr[D(H_n^{i+1})=1]$。最后根据结论 3.3.7.1，可得

$$\begin{aligned}
s_A(n)&=\frac{1}{2}+\frac{1}{2n}\cdot\sum_{i=0}^{n-1}(\Pr[D(H_n^{i+1})=1]-\Pr[D(Z)=1])\\
&=\frac{1}{2}+\frac{1}{2n}\cdot\sum_{i=0}^{n-1}(\Pr[D(H_n^{i+1})=1]\\
&\quad(2\Pr[D(H_n^i)=1]-\Pr[D(H_n^{i+1})=1]))\\
&=\frac{1}{2}+\frac{1}{n}\cdot\sum_{i=0}^{n-1}(\Pr[D(H_n^{i+1})=1]-\Pr[D(H_n^i)=1])\\
&\geq\frac{1}{2}+\frac{1}{p(n)\cdot n}
\end{aligned}$$

结论成立。□

因为 A 是多项式时间算法，所以结论 3.3.7.2 与 $\{X_n\}_{n\in\mathbb{N}}$ 是多项式时间内不可预测的前提相矛盾，从而定理的充分性得证。■

注释　展开“充分性”的证明过程，值得注意的是，其中考虑的所有混合实际上都是多项式时间内不可分辨的，因此它们都是伪随机的。此结论其实也证明了：如果 H_n^{i+1} 的 i 比特前缀是伪随机的，且 H_n^{i+1} 的$(i+1)$比特前缀是不可预测的（也可以说 H_n^{i+1} 是不可预测的），那么 H_n^{i+1} 的$(i+1)$比特前缀也是伪随机的。这就与我们在证明“充分性”之初提出的动机讨论相矛盾。

3.3.6　伪随机发生器隐含着单向函数

至此，我们已经解决了伪随机发生器是否存在的问题。在下结论之前，我们提醒大家注意的是，伪随机发生器存在的必要条件是单向函数的存在。提前说明一下：这个必要条件也是充分的；因此，伪随机发生器存在当且仅当单向函数存在。这里我们将只证明伪随机发生器存在隐含着单向函数存在。也即：

命题 3.3.8：设 G 为扩展因子为 $l(n)=2n$ 的伪随机发生器，定义函数 $f:\{0,1\}^*\to\{0,1\}^*$ 对每个 $|x|=|y|$ 都有 $f(x,y)\stackrel{\text{def}}{=}G(x)$，则 f 为强单向函数。

证明：显然，f 是多项式时间内可计算的。剩下的就是要证明每个概率多项式时间算法只能以可忽略不计的概率对 f 求逆。我们利用可约性理论。相反地，假设 A 是这样一个概率多项式时间算法，对无穷多个 n，当输入 $f(U_{2n})$时此算法能对 f 求逆的概率至少是$\frac{1}{\text{poly}(n)}$。我们将构造一个概率多项式时间算法 D，使得输入这些 n 时可以区分 U_{2n}和 $G(U_n)$，从而得到矛盾。

分辨器 D 把求逆算法 A 作为子程序。当输入 $\alpha\in\{0,1\}^*$ 时，算法 D 利用 A 获得 α 在 f 作用下的原像。接着 D 检查从 A 得到的字符串是否确实是原像并停止，如果是原像，则输出 1（反之输出 0）。即，算法 D 计算 $\beta\leftarrow A(\alpha)$，如果 $f(\beta)=\alpha$ 则输出 1，反之输出 0（即 $D(\alpha)=1$ iff $f(A(\alpha))=\alpha$）。

根据我们的假设，对某一多项式 $p(\cdot)$和无穷多个 n，有

$$\Pr[f(A(f(U_{2n})))=f(U_{2n})]>\frac{1}{p(n)}$$

根据 f 的定义，随机变量 $f(U_{2n})$等于 $G(U_n)$，因此 $\Pr[D(G(U_n))=1]=\Pr[f(A(G(U_n)))=G(U_n)]>\frac{1}{p(n)}$。另一方面，根据 f 的定义可得，至多有 2^n 个不同的 $2n$ 比特长的字符串有 f 作用下的原像。因此，$\Pr[D(U_{2n})=1]=\Pr[f(A(U_{2n}))=U_{2n}]\le 2^{-n}$。这就证明了：对无穷多个 n，有

$$\Pr[D(G(U_n))=1]-\Pr[D(U_{2n})=1]>\frac{1}{p(n)}-\frac{1}{2^n}>\frac{1}{2p(n)}$$

这与 G 的伪随机性相矛盾。■

3.4　基于单向置换的构造

本节我们介绍基于单向置换的伪随机发生器的构造。第一种方法略显抽象，它只用了一个单一的等长度的 1-1 单向函数（即，一个单一的单向置换）。第二种方法使用了同样优秀的思想，介绍了基于单向置换集合的伪随机发生器的构造方法。

3.4.1 基于单 ·置换的构造

我们提供相同伪随机发生器的两种可选择的描述。第一种描述中，我们提供能将 n 比特长的种子扩充为$(n+1)$比特长的串的伪随机发生器，它与构造 3.3.2 相结合，就产生了对每一个多项式 p，都能将 n 比特长的种子扩充为 $p(n)$ 比特长的串的伪随机发生器。展开这种结合就得到了可选择的另一种构造方法。这样得到的方法在本质上是非常吸引人的，而更重要的是，它还是基于单向置换集合的伪随机发生器的构造方法的热身准备(将在 3.2.4 小节中介绍)。

3.4.1.1 首选的描述

由定理 3.3.3(在 3.3.2 小节中介绍)，可以构造能将 n 比特长的种子扩充为$(n+1)$比特长的串的伪随机发生器。假设存在单向置换(即，1-1 的等长度的函数)，这样的伪随机发生器就很容易构造。在这里我们提醒读者，单向置换存在就意味着具有相应的核心断言的单向置换存在(见定理 2.5.2)。因此，可以证明下面的定理，其中 x、y 表示串 x 和 y 的串联(极联)。

定理 3.4.1：设 f 是等长度的 1-1 的(强单向)函数，b 是 f 的核心断言，则通过 $G(s)\stackrel{\text{def}}{=}f(s)\cdot b(s)$定义的算法 G 是一个伪随机发生器。

显然，G 是多项式时间可计算的，且 $|G(s)|=|f(s)|+|b(s)|=|s|+1$。直观地，总体 $\{f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$是伪随机的，否则，$b(U_n)$就能由 $f(U_n)$有效地推断(这与假设矛盾)。证明仅仅是对这种直觉的正式化。

实际上，我们提供两种可选择的证明。第一种证明调用定理 3.3.7(指出多项式时间不可预测性隐含了伪随机性)，从而也就限定了它的结论是总体$\{G(U_n)\}_{n\in\mathbb{N}}$是多项式时间内不可预测的。第二种证明直接建立了总体$\{G(U_n)\}_{n\in\mathbb{N}}$的伪随机性，但在说明时用到了定理 3.3.7 证明过程中的思想。

定理 3.4.1 的第一种证明：由定理 3.3.7(特别地，多项式时间不可预测性隐含了伪随机性)，可以推出总体$\{G(U_n)=f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$是多项式时间内不可预测的。

因为 f 是 1-1 且等长度的，所以随机变量 $f(U_n)$服从$\{0,1\}^n$ 上的均匀分布。因此不管计算时间有多长，$f(U_n)\cdot b(U_n)$前 n 位的每比特都不能以大于$\frac{1}{2}$的概率被预测(因为这些位相互独立并且都服从$\{0,1\}$上的均匀分布)。在指数时间内能预测(实际能确定)的只有 $f(U_n)\cdot b(U_n)$的第 $n+1$ 位(即 $b(U_n)$)。然而，由假设 b 是 f 的核心，这个位(即 $b(U_n)$)就不能由 n 位前缀(即 $b(U_n)$)在多项式时间内预测。下面是更严格的证明。

我们应用可约性理论。与我们的声明相反，假设存在有效算法 A，当输入$(1^{n+1},G(U_n))$时，能以远大于$\frac{1}{2}$的概率读 $G(U_n)$的前缀，并预测它的下一位，记为 $\text{next}_A(G(U_n))$。也就是说，对于一些正多项式 p 和无限多的 n，都有

$$\Pr[A(1^{n+1},G(U_n))=\text{next}_A(G(U_n))]>\frac{1}{2}+\frac{1}{p(n)} \tag{3.11}$$

不失一般性，我们首先声明，算法 A 总是试着猜测 $G(U_n)$的最后一位(即第 $n+1$ 位)。通过观察任意算法猜测 $G(U_n)$的任意其他位的成功概率以$\frac{1}{2}$为上限，可以证明我们的声明是合

理的。另一方面，通过随机的无偏掷币能够以$\frac{1}{2}$的成功概率猜测$G(U_n)$的任意位(特别是最后的位)。

前述结论的严密证明：给定上述算法A，我们考虑修正算法A'，步骤如下：当输入$(1^{n-1},\alpha)$时，这里$\alpha\in\{0,1\}^{n+1}$，算法A'模拟算法A的行为，总是读α的前n个位，从不读最后一位。在模拟过程中，恰能出现下面3种情况中的一种：

1. 当算法A预测α前n位的某一位时，算法A'输出一个均匀选择的位。
2. 当算法A预测α的最后一位时，算法A'输出A的预测结果。
3. 当A读α的全部位时，算法A'输出一个均匀选择的位(这里我们强调指出A'永远不能读α的最后一位)。

我们指出在1和3两种情况下，算法A的成功概率至多为$\frac{1}{2}$(且如果A输出一个位时，成功概率恰为$\frac{1}{2}$)。则在这两种情况下，A'的运行也保证了$\frac{1}{2}$的成功概率(在猜测α的最后一位时)。这样，A'的成功概率就不比A的成功概率小(在证明的最后，我们认为A和A'等同)。

接下来，我们利用算法A从$f(U_n)$预测$b(U_n)$。前面我们有$G(x)=f(x)\cdot b(x)$，这里$x\in\{0,1\}^n$。因此，由前面的结论，当输入$(1^{n+1},f(x)\cdot b(x))$时，算法A总是先读$f(x)$再猜测$b(x)$(且从来不读$b(x)$)。这样A实际上就是由$f(U_n)$预测$b(U_n)$。为了使最后的陈述更严密一些，仍需对算法做微小的修改：我们考虑算法A''，当输入$y=f(x)$时，其中$x\in\{0,1\}^n$，算法A''调用输入$(1^{n+1},y0)$时的算法A，并输出A的全部行为。因为A从来不读它的输入的最后一位，所以它的行为就与最后一位的值无关(即$A(1^{n+1},y0)\equiv A(1^{n+1},y1)$)。将这一事实与$A$总是预测它的输入的最后一位这一事实相结合(从而$\text{next}_A(y\cdot\sigma)=\sigma$)，我们有

$$\begin{aligned}\Pr[A''(f(U_n))=b(U_n)] &= \Pr[A(1^{n+1},f(U_n)\cdot 0)=b(U_n)]\\ &= \Pr[A(1^{n+1},f(U_n)\cdot b(U_n))=b(U_n)]\\ &= \Pr[A(1^{n+1},f(U_n)\cdot b(U_n))=\text{next}_A(f(U_n)\cdot b(U_n))]\end{aligned}$$

将它与方程式3.11相结合，我们有：对无限多的n，都有$\Pr[A''(f(U_n))=b(U_n)]\geq\frac{1}{2}+\frac{1}{p(n)}$，这与$b$是$f$的核心相矛盾。从而定理得证。■

定理3.4.1的第二种证明：回想前面我们有$G(U_n)=f(U_n)\cdot b(U_n)$，而我们的目标是证明总体$\{G(U_n)\}_{n\in\mathbb{N}}$和$\{U_{n+1}\}_{n\in\mathbb{N}}$是多项式时间不可分辨的。首先指出，$f(U_n)\cdot b(U_n)$的$n$位长的前缀服从$\{0,1\}^n$上的均匀分布。因此，令$\bar{b}(x)\stackrel{\text{def}}{=}1-b(x)$，我们所要做的就是证明总体$E^{(1)}\stackrel{\text{def}}{=}\{f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$和$E^{(2)}\stackrel{\text{def}}{=}\{f(U_n)\cdot\bar{b}(U_n)\}_{n\in\mathbb{N}}$是多项式时间不可分辨的(因为$\{U_{n+1}\}_{n\in\mathbb{N}}$与以$\frac{1}{2}$的概率取$E^{(1)}$，以$\frac{1}{2}$的概率取$E^{(2)}$的总体服从相同的分布)。

前述结论的进一步验证：首先注意到$E^{(1)}$与$\{G(U_n)\}_{n\in\mathbb{N}}$是等同的。再注意到$\{U_{n+1}\}_{n\in\mathbb{N}}$与总体$\{f(U_n)\cdot U_1\}_{n\in\mathbb{N}}$是同分布的，这里$U_n$与$U_1$是相互独立的随机变量。考虑到$U_1$服从$\{b(U_n),\bar{b}(U_n)\}$上的均匀分布，我们观察$f(U_n)\cdot U_1$服从这样的分布，即以$\frac{1}{2}$的概率令$E_n^{(1)}\stackrel{\text{def}}{=}f(U_n)\cdot b(U_n)$，以$\frac{1}{2}$的概率令$E_n^{(2)}\stackrel{\text{def}}{=}f(U_n)\cdot\bar{b}(U_n)$而得到的分布。这样，对每一个算法$D$，有

$$\begin{aligned}\Pr[D(U_{n+1})=1] &= \Pr[D(f(U_n)\cdot U_1)=1]\\ &= \frac{1}{2}\cdot(\Pr[D(E_n^{(1)})=1]+\frac{1}{2}\cdot\Pr[D(E_n^{(2)})=1]\end{aligned}$$

从而有

$$\begin{aligned}
&\Pr[D(G(U_n))=1]-\Pr[D(U_{n+1})=1]\\
&=\Pr[D(E_n^{(1)})=1]-\left(\frac{1}{2}\cdot\Pr[D(E_n^{(1)})=1]+\frac{1}{2}\cdot\Pr[D(E_n^{(2)})=1]\right)\\
&=\frac{1}{2}\cdot(\Pr[D(E_n^{(1)})=1]-\Pr[D(E_n^{(2)})=1])
\end{aligned}$$

因此，要想说明算法 D 不能分辨总体 $\{G(U_n)\}_{n\in\mathbb{N}}$ 和 $\{U_{n+1}\}_{n\in\mathbb{N}}$，必须首先说明 D 不能分辨总体 $E^{(1)}=\{f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$ 和 $E^{(2)}=\{f(U_n)\cdot\bar{b}(U_n)\}_{n\in\mathbb{N}}$。

下面我们证明总体 $E^{(1)}=\{f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$ 和 $E^{(2)}=\{f(U_n)\cdot\bar{b}(U_n)\}_{n\in\mathbb{N}}$ 是多项式时间不可分辨的。我们通过简化定理 3.3.7 的证明方法来证。也就是利用可分辨 $E^{(1)}=\{f(U_n)\cdot b(U_n)\}_{n\in\mathbb{N}}$ 和 $E^{(2)}=\{f(U_n)\cdot\bar{b}(U_n)\}_{n\in\mathbb{N}}$ 的任意算法（记为 D），我们构造一个基于 $f(U_n)$ 的对 $b(U_n)$ 的预测器（记为 A）。用反证法，不失一般性，我们假设对某些多项式 p 和无限多的 n，有

$$\Pr[D(f(U_n)\cdot b(U_n))=1]-\Pr[D(f(U_n)\cdot\bar{b}(U_n))=1]>\frac{1}{p(n)} \tag{3.12}$$

利用算法 D 作为一个子程序，我们构造算法 A 如下。当输入 $y=f(x)$ 时，算法 A 如下运行：

1. 在 $\{0,1\}$ 中均匀选择 σ。
2. 如果 $D(y\cdot\sigma)=1$，则输出 σ，否则输出 $1-\sigma$。

然后设 U_1 和 U_n 相互独立（这里 U_1 表示算法 A 在第 1 步选择的 σ），我们有

$$\begin{aligned}
&\Pr[A(f(U_n))=b(U_n)]\\
&=\Pr[D(f(U_n)\cdot U_1)=1\,\&\,U_1=b(U_n)]\\
&\quad+\Pr[D(f(U_n)\cdot U_1)=0\,\&\,1-U_1=b(U_n)]\\
&=\Pr[D(f(U_n)\cdot b(U_n))=1\,\&\,U_1=b(U_n)]\\
&\quad+\Pr[D(f(U_n)\cdot\bar{b}(U_n))=0\,\&\,U_1=\bar{b}(U_n)]\\
&=\frac{1}{2}\cdot\Pr[D(f(U_n)\cdot b(U_n))=1]+\frac{1}{2}\cdot(1-\Pr[D(f(U_n)\cdot\bar{b}(U_n))=1])\\
&=\frac{1}{2}+\frac{1}{2}\cdot(\Pr[D(f(U_n)\cdot b(U_n))=1]-\Pr[D(f(U_n)\cdot\bar{b}(U_n))=1])\\
&>\frac{1}{2}+\frac{1}{2p(n)}
\end{aligned}$$

这里不等式是由方程式 3.12 得来的，但这与定理的前提即 b 是 f 的核心相矛盾。■

3.4.1.2 另一种叙述

结合定理 3.3.3 和定理 3.4.1，我们可以得到，对任意多项式延拓函数 p，伪随机发生器可以将 n 比特长的种子延拓为 $p(n)$ 比特长的伪随机序列。展开这种结合，我们得到如下构造方法。

构造 3.4.2：设 $f:\{0,1\}^*\to\{0,1\}^*$ 是一个 1-1 的等长度的多项式时间可计算的函数。$b:\{0,1\}^*\to\{0,1\}$ 是多项式时间可计算的断言，$p(\cdot)$ 是任意多项式，且满足 $p(n)>n$。定义 $G(s)=\sigma_1\cdots\sigma_{p(|s|)}$，这里 $s_0\stackrel{\text{def}}{=}s$，且对每一个 $1\leq j\leq p(|s|)$，都有 $\sigma_j=b(s_{j-1})$ 和 $s_j=f(s_{j-1})$。也就是

```
Let s_0 = s and n = |s|.
For j = 1 to p(n), do
σ_j ← b(s_{j-1}) and s_j ← f(s_{j-1}).
Output σ_1σ_2···σ_{p(n)}.
```

构造方法由图 3.4 描述。注意，σ_j 很容易由 s_{j-1} 计算，但是如果 b 是 f 的核心，就有 $\sigma_j=b(s_{j-1})$ 很"难"由 $s_j=f(s_{j-1})$ 来近似。算法 G 具有伪随机性是因为 G 不能输出中间的 s_j（通过检查下面的证明过程，读者很容易就能够证实输出最后的元素（即 $s_{p(n)}$）不会影响它的伪随机性；比较命题 3.4.6）。

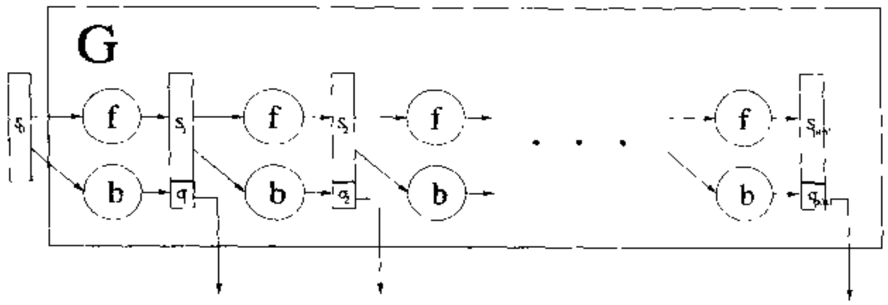


图 3.4　构造 3.4.2，作用于种子 $s_0\in\{0,1\}^n$

命题 3.4.3：设 f、b 和 G 与构造 3.4.2 中一样。如果 b 是 f 的核心，G 就是一个伪随机发生器。

证明：考虑通过将 G 的输出位反序得到的发生器 G'。也就是说，如果 $G(s)=\sigma_1\sigma_2\cdots\sigma_{p(|s|)}$，则 $G'(s)=\sigma_{p(|s|)}\cdots\sigma_1\sigma_2$。我们首先证明总体 $\{G'(U_n)\}_{n\in\mathbb{N}}$ 是伪随机的当且仅当总体 $\{G'(U_n)\}_{n\in\mathbb{N}}$ 是伪随机的。利用定理 3.3.7，能够说明总体 $\{G'(U_n)\}_{n\in\mathbb{N}}$ 是多项式时间内不可预测的。这一点是通过概括定理 3.4.1 的第一种证明方法得来的。为达到这一目的，注意下面这一等式是很有意义的。

$$G'(s)=b(f^{p(|s|)-1}(s))\cdot b(f^{p(|s|)-2}(s))\cdots b(s)$$

其中 $f^0(s)=s$，且 $f^{i+1}(s)=f^i(f(s))$。也就是说，$G'(s)$ 的第 j 位，等于 $G(s)$ 的第 $p(|s|)-j+1$ 位，即 $b(f^{p(|s|)-j}(s))$。

直观地，伪随机性的证明如下进行。用反证法，我们假设对某些 $j<t\overset{\text{def}}{=}p(n)$，给定 $G'(U_n)$ 的 j 位长的前缀，算法 A' 就能预测 $G'(U_n)$ 的第 $j+1$ 位。也就是说，给定 $b(f^{t-1}(s))\cdots b(f^{t-j}(s))$，算法 A' 可以预测 $b(f^{t-(j+1)}(s))$，其中 s 服从 $\{0,1\}^n$ 上的均匀分布。则对于服从 $\{0,1\}^n$ 上均匀分布的 x，给定 $y=f(x)$，可以调用输入为 $b(f^{j-1}(y))\cdots b(y)=b(f^j(x))\cdots b(f(x))$ 时的算法 A' 来预测 $b(x)$，而 $b(x)$ 反过来又是通过 $y=f(x)$ 在多项式时间内可计算的。在分析过程中，我们假设 f 可以导出 $\{0,1\}^n$ 上的一个置换，并结合应用 x 与 $f^{t-(j+1)}(s)$。具体如下。

用反证法，假设存在概率多项式时间算法 A' 和多项式 P'，使得对于无限多的 n，有

$$\Pr[A'(1^{p(n)},G'(U_n))=\text{next}_{A'}(G'(U_n))]>\frac{1}{2}+\frac{1}{p'(n)} \tag{3.13}$$

这样，通过构造算法 A''，当给定 $f(U_n)$ 时，能以大于 $\frac{1}{2}$ 的概率预测 $b(U_n)$，这样就得到一个矛盾。算法 A'' 运行如下，当输入 $y\in\{0,1\}^n$ 时，其中 $t\overset{\text{def}}{=}p(n)$。

1. 均匀选择 $j\in\{0,\ldots,t-1\}$。

2. 计算 $\alpha \leftarrow b(f^{-1}(y))\cdots b(y)$（注意这里 $|\alpha|=j$）。

3. 均匀选择 $\beta \in \{0,1\}^{t-j}$。

4. 当输入 $(1^t, \alpha\beta)$ 时，调用算法 A' 并记录下面的值：

(a) 在变量 l 中，记录 A' 读的 $\alpha\beta$ 前缀的长度。

(b) 在变量 τ 中，记录 A' 的输出。

5. 如果 $l=j$，输出 τ 并终止。

6. 否则（即 $l \neq j$），输出一个均匀选择的位。

显然，A'' 是在概率多项式时间内可实现的。下面我们分析当给定 $f(U_n)$ 时，A'' 预测 $b(U_n)$ 的成功概率。关键是观测输入 $f(U_n)$ 时，对于第一步中给定的每一个可能的 j 值，α 的取值（步骤 2 中 A' 所决定的）服从与分布 $G'(U_n)$ 的 j 位长的前缀相同的分布。这是因为 f 导出 $\{0,1\}^n$ 的一个置换，从而使得 $b(f^{-1}(U_n))\cdots b(U_n)$ 与 $b(f^{-1}(U_n))\cdots b(f^{-j}(U_n))$ 服从相同的分布。我们利用下面的表示法和观察资料：

● 设 R_j 是一个随机过程，给定 y 时，输出 $b(f^{-1}(y))\cdots b(y)\cdot r$，这里 r 服从 $\{0,1\}^{t-j}$ 上的均匀分布。

注意，当输入 y 时，第 1 步选择 j 后，算法 A'' 调用输入为 $(1^t, R_j(y))$ 时的 A'。通过上述"关键的"观察，$R_j(f(U_n))$ 的 j 位长的前缀与 $G'(U_n)$ 的 j 位长的前缀服从相同的分布。我们还注意到，$b(f^{-j}(f(U_n)))\cdots b(f(U_n))\cdot b(U_n)$ 与 $G'(U_n)$ 的 $j+1$ 位长的前缀服从相同的分布，而前者是连接 $R_j(f(U_n))$ 的 j 位长的前缀与 $b(U_n)$ 得来的。

● 令 $L_{A'}(\gamma)$ 是表示 $\gamma \in \{0,1\}^t$ 的前缀的长度的随机变量，当输入 $(1^t, \gamma)$ 时 A' 读它。

注意，输入 $(1^t, \gamma)$ 时，A' 的行为只依赖于 γ 的前 $L_{A'}(\gamma)$ 个位（而与 γ 的后 $t-L_{A'}(\gamma)$ 个位无关）。另一方面，$\text{next}_{A'}(\gamma)$ 又等于 γ 的第 $L_{A'}(\gamma)+1$ 位。

● 令 J 是一个表示在步骤 1 中的随机选择的随机变量，U_1 表示在步骤 6 中的随机选择的随机变量。前面我们提过，U_1 是服从 $\{0,1\}$ 上均匀分布的随机变量，与其他任何随机变量都独立。

注意，如果 $L_{A'}(\gamma)=J$，则 A'' 输出 $A'(1^t, \gamma)$ 的值，否则 A'' 输出 U_1。

利用上述几点，我们有

$$
\begin{aligned}
&\Pr[A''(f(U_n)) = b(U_n)] \\
&\quad = \Pr[A'(1^t, R_J(f(U_n))) = b(U_n) \,\&\, L_{A'}(R_J(f(U_n))) = J] \\
&\qquad + \Pr[U_1 = b(U_n) \,\&\, L_{A'}(R_J(f(U_n))) \neq J] \\
&\quad = \Pr[A'(1^t, G'(U_n)) = \text{next}_{A'}(G'(U_n)) \,\&\, J = L_{A'}(G'(U_n))] \\
&\qquad + \Pr[J \neq L_{A'}(G'(U_n))] \cdot \frac{1}{2}
\end{aligned}
$$

这里我们应用了这样一个事实，当 $L_{A'}(R_J(f(U_n)))=J$ 时，A' 的行为就只依赖于 $R_J(f(U_n))$ 的 J 位长的前缀，而后者又与 $G'(U_n)$ 的 J 位长的前缀服从相同的分布。接下来，我们还用到了下面的附加观察。

● 事件 $A'(1^t, G'(U_n)) = \text{next}_{A'}(G'(U_n))$ 与事件 J 独立。因此

$$
\begin{aligned}
&\Pr[A'(1^t, G'(U_n)) = \text{next}_{A'}(G'(U_n)) \,\&\, J = L_{A'}(G'(U_n))] \\
&\qquad = \Pr[J = L_{A'}(G'(U_n))] \cdot \Pr[A'(1^t, G'(U_n)) = \text{next}_{A'}(G'(U_n))]
\end{aligned}
$$

● 不失一般性，我们可以假设 A' 从不读它全部的输入（因为任意算法 A' 的成功概率都可

以通过修正的不读最后一位的算法 A'满足；见习题 20）。从而，$L_{A'}(G'(U_n))\in\{0,\ldots,t-1\}$ 且 $\Pr[J=L_{A'}(G'(U_n))]=\frac{1}{t}$。

将上述事实与方程式 3.13 相结合（以及 $t=p(n)$），我们有

$$\begin{aligned}\Pr[A''(f(U_n))=b(U_n)]&=\frac{1}{t}\cdot\Pr[A'(1^t,G'(U_n))=\text{next}_{A'}(G'(U_n))]+\frac{t-1}{t}\cdot\frac{1}{2}\\&\geq\frac{1}{p(n)}\cdot\left(\frac{1}{2}+\frac{1}{p'(n)}\right)+\left(1-\frac{1}{p(n)}\right)\cdot\frac{1}{2}\\&=\frac{1}{2}+\frac{1}{p(n)\cdot p'(n)}\end{aligned}$$

对无限多的 n 都成立，这与 b 是 f 的核心相矛盾。■

3.4.2 基于置换集合的构造

为了介绍基于单向置换集合的伪随机发生器的构造方法，我们应用构造 3.4.2 的基本思想。下面的一般构造是以常用的单向置换的候选为例来说明的；详细说明在抽象的陈述之后介绍。

3.4.2.1 一个抽象的陈述

设三元组(I,D,F)是一个定义单向置换集合的算法（见 2.4.2 小节），使得对 I 中的每一个 i，$D(i)$服从 f_i 定义域上的均匀分布。设 q 是一个约束算法 I 和 D 的掷币次数的多项式界（作为输入长度的函数）。[①] 对 $r\in\{0,1\}^{q(n)}$，当输入 1^n 且掷币为 r 时，我们用 $I(1^n,r)\in\{0,1\}^n$ 表示算法 I 的输出。类似地，用 $D(i,s)$表示输入为 i，掷币为 $s\in\{0,1\}^{q(n)}$时，算法 D 的输出。我们提醒读者定理 2.5.2（核心断言的存在性）同样适用于单向置换集合。

构造 3.4.4：设三元组(I,D,F)是一个定义单向置换集合的算法，B 是这个集合的核心断言。设 $p(\cdot)$是一个任意多项式。对于 $n\in\mathbb{N}$ 和 $r,s\in\{0,1\}^{q(n)}$，定义 $G(r,s)=\sigma_1\cdots\sigma_{p(n)}$，其中 $i\stackrel{\text{def}}{=}I(1^n,r)$，$s_0\stackrel{\text{def}}{=}D(i,s)$，且对每一个 $1\leq j\leq p(|s|)$，都有 $\sigma_j=B(i,s_{j-1})$，$s_j=f_i(s_{j-1})$。也就是说，当输入一个种子$(r,s)\in\{0,1\}^{q(n)}\times\{0,1\}^{q(n)}$时，算法 G 如下运行，其中 $F(i,x)=f_i(x)$。

```
Set i←I(1^n,r) and s_0←D(i,s).
For j=1 to p(n), do
    σ_j←B(i,s_{j-1}) and s_j←F(i,s_{j-1}).
Output σ_1σ_2…σ_{p(n)}.
```

当输入种子(r,s)时，算法 G 首先应用 r 来确定 D_i上的 f_i（即 $i\leftarrow I(1^n,r)$）。然后，算法 G 应用 s 来确定均匀分布在 D_i上的"起点"s_0。算法 G 的主要部分就是重复应用 f_i 作用于起点 s_0，并输出每个结果元素的核心断言。这一部分模拟构造 3.4.2，只不过是用预先确定的置换 f_i 代替了那里的单一置换 f。算法 G 的扩充性质依赖于多项式的 $p(\cdot)$选择，即多项式应该比多项式 $q(\cdot)$的两倍还大。

定理 3.4.5：设(I,D,F)，B，$q(\cdot)$，$p(\cdot)$和 G 与构造 3.4.4 中的一样，并假设对所有的

① 在许多情况下，多项式 q 是线性的。实际上，我们可以修改单向置换的任意集合，使得 $q(n)=n$；见第 2 章练习 19。

n、有 $p(n)>2q(n)$。进一步假设，对于算法 I 中的每一个 i，随机变量 $D(i)$ 服从集合 Di 上的均匀分布。则 G 是一个伪随机发生器。

定理 3.4.5 是下述命题的直接推论。

命题 3.4.6：设 $n,t\in Z$。对每一个 $I(1^n)$ 中的 i 和每一个 $x\in D_i$，定义

$$G_i^t(x)=B(i,x)\cdot B(i,f_i(x))\cdots B(i,f_i^{t-1}(x))$$

这里对任意 $j\geq 0$，有 $f_i^0(x)=x, f_i^{j+1}(x)=f_i^j(f_i(x))$。设 (I,D,F) 和 B 与定理 3.4.5 中的一样，令 I_n 是表示 $I(1^n)$ 的一个随机变量，$X_n=D(I_n)$ 表示均匀分布于 D_{I_n} 上的随机变量。那么对每一个多项式 $p(\cdot)$，总体

$\{(I_n,G_{I_n}^{p(n)}(X_n),f_{I_n}^{p(n)}(X_n))\}_{n\in\mathbb{N}}$ 和 $\{(I_n,U_{p(n)},f_{I_n}^{p(n)}(X_n))\}_{n\in\mathbb{N}}$

是多项式时间不可分辨的。

因此，这个著名的算法不仅检测了 $p(n)$ 位长的序列，还得到了算法 G 选择的指标 i（G 的计算的第一步）和由 G 计算的最后一个域元素（即 $f_i^{p(n)}(X_n)$）。即使有这样的附加信息，要分辨 $G_{I_n}^{p(n)}(X_n)\equiv G(U_{2q(n)})$ 和 $U_{p(n)}$ 也是不可行的。我们注意到，为这个著名算法提供 $f_i^{p(n)}(X_n)$ 只能使命题更严格一些，而这种严格形式对定理 3.4.5 的证明是不必要的。但是，这种严格形式将在第 5 章用到。

略证：证明与命题 3.4.3 的证明类似。特别地，我们令 $\overline{G}_i^t(x)=B(i,f_i^{t-1}(x))\cdots B(i,x)$（$G_i^t(x)$ 的逆元），并证明即使给定 I_n 和 $f_{I_n}^{p(n)}(X_n)$ 作为辅助输入，序列 $\overline{G}_{I_n}^{p(n)}(X_n)$ 也是在多项式时间内不可预测的。我们通过可约性理论来证明：一个预测 $\overline{G}_{I_n}^{p(n)}(X_n)$ 的下一位的算法，如果给定 I_n 和 $f_{I_n}^{p(n)}(X_n)$，可用来构造一个新的算法，由 I_n 和 $f_{I_n}(X_n)$ 来预测 $B(I_n,X_n)$。这与我们的假设 B 是集合 (I,D,F) 的核心断言相矛盾。为了说明对每一个 $j<t$，$D(i)$ 和 $f_i^j(D(i))$ 服从相同分布[①]，我们还必须假设 $D(i)$ 服从 D_i 上的均匀分布。读者必须能完成这个证明。■

总结 即使将 $D(i)$ 的分布条件放松一些，只要求 $D(i)$ 与 D_i 上的均匀分布统计接近（是 $|i|$ 的函数），命题 3.4.6 和定理 3.4.5 仍然是成立的。同理，我们可以放松 I 的条件，使得上述结论对除了一个可忽略由 $I(1^n)$ 生成的 i 的测度外都成立（而不是对所有这样的 i）。

3.4.2.2 具体的实例

作为构造 3.4.4 的直接应用，我们得到基于下面某一假设基础上的伪随机发生器。

- 离散对数问题的难处理性：特别地，我们假设在 2.4.3 小节中介绍的 DLP 集合是单向的。发生器的依据是，在上述假设下，下面的问题是难处理的：给定一个素数 P，模 P 乘法群的一个本原元素 G，以及这个群中的一个元素 Y，猜测是否存在 $0\leq x\leq P/2$，使得 $Y\equiv G^x \bmod P$。换句话说，后面的预测（记为 B_P）构成了集合 DLP 的一个核心。

为了选择一个素数 P，模 P 乘法群的一个本原元素 G、以及这个群中的一个元素 Y，发生器使用种子，并输出序列

$$B_P(Y),B_P(G^Y \bmod P),B_P(G^{G^Y \bmod P} \bmod P),\ldots$$

也就是说，重复迭代的函数是 $Z\to G^Z \bmod P$。

- RSA 的求逆困难：特别地，我们假设在 2.4.3 小节中介绍的 RSA 集合是单向的。发生

---

① 我们指出较弱一些的假设就可以满足要求。或者我们可以假设对每一个 $j<t$，函数 f_i 当输入分布 $f_i^j(D(i))$ 时很难求逆。

器的依据是，在这种假设下，最低有效位（记为 lsb）构成 RSA 的核心。

为了选择一对素数(P,Q)，一个与$\phi(N)=(P-1)\cdot(Q-1)$互素的素数e，以及模$N\stackrel{\rm def}{=}P\cdot Q$乘法群中的一个元素$X$，发生器使用种子，并输出序列

$$\mathrm{lsb}(X),\mathrm{lsb}(X^e \bmod N),\mathrm{lsb}(X^{e^2 \bmod \phi(N)} \bmod N),\mathrm{lsb}(X^{e^3 \bmod \phi(N)} \bmod N),\ldots$$

也就是说，重复迭代的函数是$Z\to Z^e \bmod N$。

- 布卢姆（Blum）整数的分解困难：特别地，我们假设，给定两个大素数的乘积，每一个都是与 3(mod 4)同余的，再获得这两个素数是不可行的。在这种假设下，发生器的依据是最低有效位构成平方求模函数的一个核心断言。同时我们还用到这样的事实，即对这样的模数（称为布卢姆整数），平方求模运算导出二次剩余上的一个置换。

为了选择一对与 3(mod4)同余的整数(P,Q)，以及模$N\stackrel{\rm def}{=}P\cdot Q$乘群中的一个元素，发生器使用种子，并输出序列

$$\mathrm{lsb}(X),\mathrm{lsb}(X^2 \bmod N),\mathrm{lsb}(X^{2^2 \bmod \phi(N)} \bmod N),\mathrm{lsb}(X^{2^3 \bmod \phi(N)} \bmod N),\ldots$$

也就是说，重复迭代的函数是$Z\to Z^2 \bmod N$。

所有这些建议都依赖于选择随机素数的随机算法。因此，对于这样的算法使用的随机位，越少越好。生成n位长的随机素数的明显算法使用的是$O(n^3)$个随机位（见附录 A）。我们指出存在更随机有效的算法，只用$O(n)$个随机位就可以生成n位长的素数。

3.4.3* 应用核心函数而不是核心断言

构造 3.4.2（和 3.4.4）可以很容易地被一般化为具有核心函数而不是核心断言的单向置换（单向置换集合）。这种构造的优点是，对于单向置换的每一个应用程序，发生器输出位的数目比较大（即，比 1 大）。我们假设读者对定义核心函数的 2.5.3 小节很熟悉，下面我们只介绍构造 3.4.4 的一般化。

构造 3.4.7：设(I,D,F)和构造 3.4.4 中一样，并假设H是相应的核心函数。设$p(\cdot)$是一任意多项式。对每一个$n\in\mathbb{N}$和$r,s\in\{0,1\}^{q(n)}$，定义$G(r,s)=\alpha_1\cdots\alpha_{p(n)}$，其中$i\stackrel{\rm def}{=}I(1^n,r)$，$s_0\stackrel{\rm def}{=}D(i,s)$，且对每一个$1\le j\le p(|s|)$，都有$\alpha_j=H(i,s_{j-1})$和$s_j=f_i(s_{j-1})$成立。

对一个核心函数H，由$I(1^n)$产生的i，我们用$l_H(n)$表示以 2 为底$H(i,\cdot)$的定义域的测度的对数。任意的核心断言都可以看作$l_H(n)=1$时的核心函数。回想前面，任意单向函数都可以被修改，使它具有$l_H(n)=O(\log n)$的核心函数H（见定理 2.5.6）。同时，如果假设 RSA 集合是单向的，$O(\log n)$个最低有效位就构成一个核心函数（$l_H(n)=O(\log n)$）。对于 Rabin 集合有相同的结论。

定理 3.4.8：设(I,D,F)，H，$q(\cdot)$，$p(\cdot)$和G与构造 3.4.7 中一样，并假设对所有的n都有$p(n)\cdot l_H(n)>2q(n)$，进一步假设对于算法I定义域内的每一个i，随机变量$D(i)$都服从D_i上的均匀分布，则G是一个伪随机发生器。

定理的证明是通过对定理 3.4.5 证明的一个很自然的推广，在这里略去不证。再者，算法不仅检测了$p(n)\cdot l_H(n)$位长的序列，还得到G选择的指标I（在G的计算过程的第一步中进行）和由G计算的最后一个域元素（即$f_i^{p(n)}(s_0)$），定理的结论仍然成立。但是即使有这个附加的信息，要分辨$G(U_{2q(n)})$和$U_{p(n)\cdot l_H(n)}$仍然是不可行的。

对于单向集合的每一个应用程序，构造 3.4.7 中的发生器输出$l_H(n)$个位，其中H是相应

的核心函数。因此，如果能够证明对于 Rabin 集合，$l_H(n)=\Omega(n)$的核心函数 H 存在，则存在非常有效的伪随机发生器（对于一个 n 位模数的每一个模平方，产生 $\Omega(n)$个位）。

3.5* 基于单向函数的构造

我们已经知道，单向函数存在当且仅当伪随机发生器存在。然而，目前已知的把任意单向函数转变成伪随机发生器的构造都是不切实际的。而且，证明此构造确实能产生伪随机发生器的过程是非常复杂的，不适合在本书讨论。相反，我们只讨论基于此构造的一些思想以及部分结论（我们相信这些思想在某些地方是很有用的）。

3.5.1 利用 1-1 单向函数

我们已经知道如果 f 是 1-1 等长度的单向函数，b 是相应的核心断言，那么 $G(s)\stackrel{\text{def}}{=}f(s)\cdot b(s)$构成了一个伪随机发生器，其中 $x\cdot y$ 表示字符串 x 和 y 的串联。下面我们放宽对 f 的要求，假设 f 是 1-1 单向函数（但不一定是等长度的）。不失一般性，我们假设存在一个多项式 $p(\cdot)$，对所有 x 有 $|f(x)|=p(|x|)$。如果 f 不是等长度的，可得 $p(n)>n$。初看起来，你可能认为我们不只在这种情况下受益，因为 f 本身具有可扩展性。再一想，你可能会意识到这种好处并不明显，因为扩展得到的字符串可能不是“看似随机”的。特别地，可能会是这种情况：对所有 x 来说，$f(x)$的第一位都是 0。而且也可能是这样的情况：对所有 x 来说，$f(x)$的前 $|f(x)|-|x|$ 位都是 0。一般说来，$f(U_n)$可能很容易同 $U_{p(n)}$ 区分开（否则 f 本身就能构造伪随机发生器了）。因此，大多数情况下，我们必须除掉 f 的扩展性，因为它与“伪随机”性不共存。通常，我们必须让 $f(U_n)$收缩至长度近似为 n，使得收缩后的结果是均匀分布的。于是问题就归结为如何有效地实现这个收缩过程。

假设存在一个可有效计算的函数 h，使得 $f_h(x)\stackrel{\text{def}}{=}h(f(x))$是等长度和 1-1 的。在这种情况下，令 $G(s)\stackrel{\text{def}}{=}h(f(s))\cdot b(s)$（其中 b 是 f 的核心断言），这就得到一个伪随机发生器。G 的伪随机性可以利用以下结论得到：如果 b 是 f 的核心，那它也是 f_h 的核心（因为通过首先应用 h，可以很容易修改从 $h(f(x))$推测 $b(x)$的算法，使之能从 $f(x)$推测出 $b(x)$。但问题是我们对 f 的结构一无所知，因此不能保证这样的 h 存在。均匀选择的“哈希”函数具有与上述性质近似的性质。因此，哈希函数在此类构造中起核心作用，我们必须首先讨论这些函数。

3.5.1.1 哈希函数

设 S_n^m 是一个字符串的集合，这些字符串描述了一些把长为 n 的字符串映射为长为 m 的字符串的函数。为了简化，我们假设对某一函数 l 有 $S_n^m=\{0,1\}^{l(n,m)}$。从而，我们可以自由地把 S_n^m 中的字符串和它们所代表的函数相关联。设 H_n^m 是一个在集合 S_n^m 上均匀分布的随机变量。如果 S_n^m 满足以下 3 个条件，我们就把它称为一个哈希族（或哈希函数族）：

1. S_n^m 是两两独立的映射族：对于每个 $x\neq y\in\{0,1\}^n$，随机变量 $H_n^m(x)$和 $H_n^m(y)$是独立的且在$\{0,1\}^m$ 上均匀分布。

2. S_n^m 有简单表达式：$S_n^m=\{0,1\}^{\text{poly}(n,m)}$。

3. 可以有效地对 S_n^m 求值：存在一个多项式时间算法，该算法在输入一个函数 h 的表达式（属于 S_n^m）和字符串 $x\in\{0,1\}^n$ 时，返回值为 $h(x)$。

需要强调的是，这里定义的哈希族对困难性不作要求，是独立于任何棘手性假设而存在的。[①] 目前广泛使用的哈希族是把 n 维二进制向量映射成 m 维二进制向量的仿射变换（即用一个 $n\times m$ 的二进制矩阵去乘此 n 维向量，再加上一个 m 维向量就可以得到此变换）的集合。如果只考虑利用 Toeplitz 矩阵族（即对角线上的元素相等的矩阵）得到的变换，就可以获得有更多简单表达式的哈希族。详述参见习题 22。

下面这个关于哈希函数的引理是我们分析的基础（也是哈希函数在复杂度理论中的许多应用的基础）。简而言之，这个引理断言：如果一个随机变量 X_n 没有把大概率集赋给任何单个的字符串，那么对哈希族中的大多数 h，有 $h(X_n)$ 几乎是均匀分布的。特别地，当使用如前所述的哈希族 S_n^m 时，我们将只考虑对每个 $x\in\{0,1\}^n$ 满足 $\Pr[X_n=x]\ll 2^{-m}$ 的随机变量 X_n。

引理 3.5.1：设 m,n 都是整数，且 $m<n$，S_n^m 是一个哈希族，b 和 δ 是满足 $m\le b\le n$ 和 $\delta\ge 2^{-\frac{b-m}{2}}$ 的两个实数。假设 X_n 是一个在 $\{0,1\}^n$ 上分布的随机变量，使得对于每个 x，$\Pr[X_n=x]\le 2^{-b}$ 成立。那么对于每个 $\alpha\in\{0,1\}^m$，S_n^m 中最多有 $2^{-(b-m)}\delta^{-2}$ 的 h 满足

$$\Pr[h(X_n)=\alpha]\in(1\pm\delta)\cdot 2^{-m}$$

$\Pr[h(X_n)=\alpha]$ 在所有 h 上的平均值等于 2^{-m}。因此这个引理就指定了偏离此平均值的那部分 h 的上限。特别地，若函数 h 不满足 $\Pr[h(X_n)=\alpha]\in(1\pm\delta)\cdot 2^{-m}$，则它（对 α 和随机变量 X_n 来说）是无效的。此引理断言：无效函数所占的比例最多是 $2^{-(b-m)}\delta^{-2}$。典型地，我们将利用 $\delta\stackrel{\text{def}}{=}2^{-\frac{b-m}{3}}\ll 1$（使得与平均值的偏差等于无效的 h 所占的比例数）。另一种有用的选择是 $\delta\ge 1$（它的无效的 h 所占的比例甚至更小，然而这里的非无效状态仅仅意味着 $\Pr[h(X_n)=\alpha]\le(1\pm\delta)\cdot 2^{-m}$，因为 $\Pr[h(X_n)=\alpha]\ge 0$ 总是成立的）。

证明：选定任意一个满足此引理条件的随机变量 X_n 和任意的 $\alpha\in\{0,1\}^m$。记 $\omega_x\stackrel{\text{def}}{=}\Pr[X_n=x]$。对每个 h，有

$$\Pr[h(X_n)=\alpha]=\sum_x\omega_x\zeta_x(h)$$

其中若 $h(x)=\alpha$，则 $\zeta_x(h)$ 等于 1，否则等于 0。因此，我们只对在所有可能的 h 上 $|2^{-m}-\sum_x\omega_x\zeta_x(h)|>\delta 2^{-m}$ 成立的概率感兴趣。若把 ζ_x 看作是定义在随机变量 H_n^m 上的随机变量，那么接下来就是要证明

$$\Pr\left[\left|2^{-m}-\sum_x\omega_x\zeta_x\right|>\delta\cdot 2^{-m}\right]<\frac{2^{-(b-m)}}{\delta^2}$$

上式可以利用契比雪夫不等式，按如下步骤证明：

1. ζ_x 是两两独立的，且 $\text{Var}(\zeta_x)<2^{-m}$（因为 $\zeta_x=1$ 的概率为 2^{-m}，其余的 $\zeta_x=0$）。

2. （根据前提假设有）$\omega_x\le 2^{-b}$ 且 $\sum_x\omega_x=1$。

也即，

$$\Pr\left[\left|2^{-m}-\sum_x\omega_x\zeta_x\right|>\delta\cdot 2^{-m}\right]\le\frac{\text{Var}[\sum_x\omega_x\zeta_x]}{(\delta\cdot 2^{-m})^2}$$

① 相反，诸如无冲突散射和通用单向散射此类的概念必须具备困难性，而且仅当单向函数存在时存在。（无冲突散射和通用单向散射将在本书第二卷的第 6 章中定义和讨论。）

$$= \frac{\sum_x \omega_x^2 \cdot \mathsf{Var}(\zeta_x)}{\delta^2 \cdot 2^{2m}}$$

$$< \frac{2^{-m} 2^{-b}}{\delta^2 \cdot 2^{-2m}}$$

引理得证。

3.5.1.2 基本构造

利用任何 1-1 单向函数和任何哈希族，我们可以迈出构造伪随机发生器的重要一步。

构造 3.5.2：函数 $f:\{0,1\}^* \to \{0,1\}^*$ 对某一多项式 $p(\cdot)$ 和所有 x，有 $|f(x)| = p(|x|)$。对任何整形函数 $l: \mathbb{N} \to \mathbb{N}$，令 $g:\{0,1\}^* \to \{0,1\}^*$ 为满足 $|g(x)| = l(|x|)+1$ 的函数，并令 $S_{p(n)}^{n-l(n)}$ 是哈希族。对每个 $x \in \{0,1\}^n$ 和 $h \in S_{p(n)}^{n-l(n)}$，定义

$$G(x,h) \stackrel{\text{def}}{=} (h(f(x)), h, g(x))$$

显然，$|G(x,h)| = (|x| - l(|x|)) + |h| + (l(|x|)+1) = |x| + |h| + 1$。因此，$G$ 满足扩展条件。下面这个命题给出了 G 的输出和均匀总体之间的可分辨性的上界(可惜这个上界仅当 $l:\mathbb{N} \to \mathbb{N}$ 是超对数时才能忽略不计)。

命题 3.5.3：设 f, l, g 和 G 如前所述。假设 f 是 1-1 的，g 是 f 的核心函数。那么对于每个概率多项式时间算法 A、每个正多项式 $p(\cdot)$，以及所有足够大的 n，有

$$|\Pr[A(G(U_n, U_k)) = 1] - \Pr[A(U_{n+k+1}) = 1]| < 2 \cdot 2^{-\frac{l(n)}{3}} - \frac{1}{p(n)}$$

其中 k 是 $S_{p(n)}^{n-l(n)}$ 中哈希函数表达式的长度。

根据习题 22 和 23，我们可以利用 $k = O(n)$ 这一结论。特别地，上述命题对形如 $l(n) \stackrel{\text{def}}{=} c\log_2 n$ 的函数 $l(\cdot)$ 成立，其中 c 是一个大于 0 的常数。对这样的函数 l 而言，每个单向函数都(可以很容易被修改成)有一个满足命题的前提条件的核心 g(的函数)(参见 2.5.3 节)。从而，我们非常接近于构造伪随机发生器这一目标了(参见后文)。

略证 设 $H_{p(n)}^{n-l(n)}$ 为在 $S_{p(n)}^{n-l(n)}$ 上均匀分布的随机变量。我们首先注意到

$$G(U_{n+k}) \equiv (H_{p(n)}^{n-l(n)}(f(U_n)), H_{p(n)}^{n-l(n)}, g(U_n))$$

$$U_{n+k+1} \equiv (U_{n-l(n)}, H_{p(n)}^{n-l(n)}, U_{l(n)+1})$$

再考虑$(H_{p(n)}^{n-l(n)}(f(U_n)), H_{p(n)}^{n-l(n)}, U_{l(n)+1})$这一混合分布。此命题可以通过以下两个结论直接得到。

结论 3.5.3.1：总体

$$\{(H_{p(n)}^{n-l(n)}(f(U_n)), H_{p(n)}^{n-l(n)}, g(U_n))\}_{n\in\mathbb{N}}$$

和

$$\{(H_{p(n)}^{n-l(n)}(f(U_n)), H_{p(n)}^{n-l(n)}, U_{l(n)+1})\}_{n\in\mathbb{N}}$$

是多项式时间内可区别的。

证明思路：使用可约性理论。如果结论不成立，就能导出 g 是 f 的核心这一前提的矛盾。特别地，若给定一个不符合此结论的算法 D，我们构造一个算法 D'，该算法当输入(y, z)时，均匀地选择一个 $h \in S_{p(n)}^{n-l(n)}$，并输出 $D(h(y), h, z)$。那么 D' 就能分辨$\{(f(U_n), g(U_n))\}_{n\in\mathbb{N}}$和$\{(f(U_n), U_{l(n)+1})\}_{n\in\mathbb{N}}$。

结论 3.5.3.2：随机变量

$$(H_{p(n)}^{n,l(n)}(f(U_n)), H_{p(n)}^{n,l(n)}, U_{l(n)+1})$$

和

$$(U_{n-l(n)}, H_{p(n)}^{n-l(n)}, U_{l(n)+1})$$

之间的统计差异限制在 $2 \cdot 2^{-l(n)/3}$ 之内。

证明思路：利用 $S_{p(n)}^{n,l(n)}$ 是哈希族这一前提假设，并运用引理 3.5.1。特别地，利用 $\delta = 2^{-l(n)/3}$，并注意到对每个 y 有 $\Pr[f(U_n)=y] \leq 2^{-n}$，再分别计算无效的 h 和非无效的 h 对 $(H_{p(n)}^{n-l(n)}(f(U_n)), H_{p(n)}^{n-l(n)})$ 和 $(U_{n-l(n)}, H_{p(n)}^{n,l(n)})$ 之间的统计差异的作用。

因为统计差异受算法的分辨力限制，所以命题成立。

扩展 命题 3.5.3 可以扩展成函数 f 是多项式到 1(而不是 1 到 1)的情况。特别地，对某一多项式 $q(\cdot)$ 和所有足够长的 x，令 f 满足 $|f^{-1}(f(x))| < q(|x|)$。修改后的命题断言：对每个概率多项式时间算法 A、每个多项式 $p(\cdot)$ 和所有足够大的 n，有

$$|\Pr[A(G(U_n, U_k)) = 1] - \Pr[A(U_{n-k-1}) = 1]| < 2 \cdot 2^{-\frac{l(n)-\log_2 q(n)}{3}} + \frac{1}{p(n)}$$

其中 k 的定义和命题 3.5.3 中一样。

3.5.1.3 得到伪随机发生器

我们已经证明了命题 3.5.3，现在可以考虑利用它来构造伪随机发生器的可能性。需要强调的是，应用含有长度函数 $l(\cdot)$ 的命题 3.5.3 时，必须有一个 f 的核心函数 g，满足 $|g(x)| = l(|x|)+1$。根据定理 2.5.6(见 2.5.3 节)可知，倘若 $l(\cdot)$ 是对数函数，则对所有的单向函数来说，这样的核心函数实质上都存在(事实上，定理 2.5.6 可以断言：任何单向函数被修改后这样的核心函数仍然存在，其中修改后的函数保持了原始函数的 1-1 性)。因此，结合定理 2.5.6 和命题 3.5.3，利用对数长度函数，我们就几乎可以构造伪随机发生器了。特别地，对每个多项式 $p(\cdot)$，利用 $l(n) \stackrel{\text{def}}{=} 3\log_2 p(n)$，可以构造一个确定性多项式时间算法，此算法把 $O(n)$ 比特长的种子扩展成 $(O(n)+1)$ 比特长的字符串，且任何多项式时间算法都不能以大于 $\frac{1}{p(n)}$ 的概率把其输出的字符串和均匀选择的字符串区别开来(除了 n 是有限值的情况)。但是这并不意味着输出字符串就是伪随机的(即有区别的距离小于任何多项式分数)。(因为我们不能使用大于任何对数函数的 $l(\cdot)$，因此)还需要一个附加概念。下面，我们将提出两种通过构造 3.5.2 得到伪随机发生器的方法。

第一种方法 首先我们利用(3.3.2 节中的)构造 3.3.2 来提高从构造 3.5.2 得到的算法的扩展因数。特别地，对每个 $i \in \mathbb{N}$，我们构造一个确定性多项式时间算法，记为 G_i，此算法把 n 比特长的种子扩展成 n^3 比特长的字符串，且任何多项式时间算法都不能以大于 $\frac{1}{n^i}$ 的概率把其输出的字符串和均匀选择的字符串区别开来(除了 n 是有限值的情况)。这些算法记为 G_1，G_2，…。现在我们可以按如下方式构造一个伪随机发生器 G：令

$$G(s) \stackrel{\text{def}}{=} G_1(s_1) \oplus G_2(s_2) \oplus \cdots \oplus G_{m(|s|)}(S_{m(|s|)})$$

其中，$\oplus$ 表示字符串间逐位异或(XOR)，$s_1 s_2 \cdots s_{m(|s|)} = s$，$|s_i| = \frac{|s|}{m(|s|)} \pm 1$，$m(n) \stackrel{\text{def}}{=} \sqrt[3]{n^4}$①。显然，$|G(s)| \approx (\frac{|s|}{m(|s|)})^3 = |s|^2$。$G$ 的伪随机性可以通过可约性理论来证明。特别地，如果对

---

① 可相当任意地选择函数 $m: \mathbb{N} \to \mathbb{N}$；任何满足 $m(n) < n^{2/3}$ 的无界函数 $m: \mathbb{N} \to \mathbb{N}$ 都可以。

于某一 i 和无穷多个 n，某一多项式时间算法能够以大于$\frac{1}{n^{2c+3}}$的概率分辨 $G(U_n)$ 和 U_{n^2}，那么我们就能（在多项式时间内）以大于$\frac{1}{n^{2c+3}}=\frac{1}{(n'/m(n))^{c}}$的概率分辨 $G_i(U_{n'/m(n)})$ 和 $U_{(n'/m(n))^3}$，这与我们关于 G_i 的假设相矛盾。

第二种方法　这里我们把构造 3.5.2 作用于按如下方式定义的函数 $\overline{f}$：

$$\overline{f}(x_1,\cdots,x_n)\stackrel{\text{def}}{=}f(x_1)\cdots f(X_n)$$

其中 $|x_1|=\cdots=|x_n|=n$。把构造 3.5.2 作用于函数 $\overline{f}$ 的好处是，它使我们能使用 $l(n^2)\stackrel{\text{def}}{=}n-1$，这样命题 3.5.3 就能说明 G 是伪随机发生器。现在剩下的就是要证明 $\overline{f}$ 有一个能把长为 n^2 的字符串映射成长为 n 的字符串的核心函数。假设 b 是函数 f 的核心断言，我们可以为 $\overline{f}$ 构造这样一个核心函数。特别地：

构造 3.5.4：设 $f:\{0,1\}^*\rightarrow\{0,1\}^*$，$b:\{0,1\}^*\rightarrow\{0,1\}$。定义

$$\overline{f}(x_1,\ldots,x_n)\stackrel{\text{def}}{=}f(x_1)\cdots f(x_n)$$

$$\overline{g}(x_1,\ldots,x_n)\stackrel{\text{def}}{=}b(x_1)\cdots b(x_n)$$

其中 $|x_1|=\cdots=|x_n|=n$。

命题 3.5.5：f 和 b 如构造 3.5.4 所述。若 b 是 f 的核心断言，则 $\overline{g}$ 是 $\overline{f}$ 的核心断言。

证明思路：利用混合技术。第 i 个混合是

$$(f(U_n^{(1)}),\ldots,f(U_n^{(n)}),b(U_n^{(1)}),\ldots,bU_n^{(i)}),U_1^{(i+1)},\ldots,U_1^{(n)})$$

当然，第 n 个混合就等于$(\overline{f}(U_{n^2}),\overline{g}(U_{n^2}))$，其中第 0 个混合等于$(\overline{f}(U_{n^2}),U_n)$。接着说明如何把一个区分相邻混合的算法转化成一个 $f(U_n)$ 的断言 $b(U_n)$。特别地，此变换类似于定理 3.3.7 的“充分性”证明过程中和定理 3.4.1 的第二种证明方法中使用的构造。

小结　利用上述任意一种方法，可得如下定理。

定理 3.5.6：如果 1-1 单向函数存在，那么伪随机发生器也存在。

所有的结论都可以扩展成函数 f 是多项式对 1（而不是 1 对 1）的情况。特别地，对某一多项式 $q(\cdot)$ 和所有充分长的 x，令 f 满足 $|f^{-1}f(x)|<q(|x|)$。我们可以断言：如果 f 是单向的，那么（上述任何一种方法都可以证明）伪随机发生器存在。倘若扩充命题 3.5.3（这在 3.5.1.2 节的末尾有说明），利用第一种方法证明后一结论是非常直接的。而要用第二种方法证明这一结论，只需把构造 3.5.2 作用于函数 $\overline{f}$，其中 $l(n^2)\stackrel{\text{def}}{=}n-1+n\cdot\log_2 q(n)$。这就需要证明 $\overline{f}$ 有一个能把长为 n^2 的字符串映射成长为 $(n\cdot(1+\log_2 q(n)))$ 的字符串的核心函数。假设 g 是函数 f 的核心函数，且 $|g(x)|=1+\log_2 q(|x|)$，我们就能为 $\overline{f}$ 构造这样一个核心函数。特别地，

$$\overline{g}(x_1,\ldots,x_n)\stackrel{\text{def}}{=}g(x_1)\cdots g(x_n)$$

其中 $|x_1|=\cdots=|x_n|=n$。

3.5.2　利用正则单向函数

命题 3.5.3 的有效性主要依赖于这样一个事实：如果 f 是 1-1 的，那么 $f(U_n)$ 在严格意义上保持 U_n 的“熵”（即对每个 α，有 $\Pr[f(U_n)=\alpha]\leq 2^{-n}$）。在这种情况下，就能缩短 $f(U_n)$（至 $n-l(n)$ 位）并得到在 $\{0,1\}^{n-l(n)}$ 上几乎均匀的分布。正如前面强调的那样，我们可以适当放宽

条件，只要求 f 是多项式对 1（而不是 1 对 1）的。此时，只会导致"熵"呈对数减少，而这可通过适当增大核心函数的范围来补偿。需要强调的是，可以构造任意单向函数的具有对数长度的核心函数。然而，函数 f 通常不是多项式对 1 的，而是能把指数多个原像映射成同一个像。若是那样，则把 f 作用于 U_n 会引起更大的"熵"减少，从而无法用前面的方法来补偿。例如，如果对 $|x|=|y|$，有 $f(x,y)\stackrel{\text{def}}{=}f'(x)O^{|y|}$，那么对某一 α，有 $\Pr[f(U_n)=\alpha]\geq 2^{-\frac{|\alpha|}{2}}$。此时就需要一种新方法，下面我们将给出这种方法。

如果 f 把不同的原像映射为同一个像 y，我们可以通过集合 $f^{-1}(y)$ 中原像的索引来增大 y，而不会影响 f 的求逆困难性。即，我们定义 $F(x)\stackrel{\text{def}}{=}f(x)\cdot \text{idx}_f(x)$，其中 $\text{idx}_f(x)$ 表示 x 在集合 $\{x':f(x')=f(x)\}$ 中的索引（按辞典顺序排列）。我们认为：对 F 求逆并非比对 f 求逆更容易。这可以用可约性理论法来证明。给定一个对 F 求逆的算法，我们可以按如下方式对 f 求逆。输入 y（假设它在 $f(V_n)$ 的值域中），首先在 $\{1,...,n\}$ 上均匀地选择一个 m，接着在 $\{1,...,2^m\}$ 上均匀地选择一个 i，最后试着对 (y,i) 求其在 F 作用下的逆。分析此算法时，应考虑 $i=\lceil\log_2|f^{-1}(y)|\rceil$ 这一情况。

我们给出的函数 F 确实保持了 f 的求逆困难性，但问题是 F 没有保持 f 的易于计算性。特别地，对一般的 f 来说，如何计算 $\text{idx}_f(x)$ 并不是很明显；我们只知道这个任务可以在指数时间（和多项式空间）内实现。而且，哈希函数有助于其实现。例如，假设 f 是作用于长为 n 的字符串上的 2^m 对 1 的函数。那么我们可以令 $\text{idx}_f(x)=(H_n^m,H_n^m(x))$，得到原像集合的"随机指标"。需要强调的是，此方法要求对（给定的像的）原像集合的大小有个较好的预测。也就是说，对于给定的 x，$|f^{-1}(f(x))|$ 是易于计算的。对正则函数来说，这种预测是比较方便的。

定义 3.5.7（正则函数）（*Regular Fanctions*）：如果对函数 $f:\{0,1\}^*\to\{0,1\}^*$ 来说，存在一个整形函数 $m:\mathbb{N}\to\mathbb{N}$，使得对于所有足够长的 $x\in\{0,1\}^*$，有

$$|\{y: f(x)=f(y)\land |x|=|y|\}|=2^{m(|x|)}$$

那么就称函数 f 是正则的。

为了简化，读者可以进一步假设存在这样一个算法：输入 n 时，此算法可以在 $\text{poly}(n)$ 时间内计算 $m(n)$。正如在本小节结尾我们将看到的那样，如果没有这一假设也是成立的。为了简化（概念），我们假设若 $f(x)=f(y)$，则 $|x|=|y|$。

构造 3.5.8：设 $f:\{0,1\}^*\to\{0,1\}^*$ 是一个正则函数，且对某一整形函数 $m(\cdot)$，有 $m(|x|)=\log_2|f^{-1}(f(x))|$。令 $L:\mathbb{N}\to\mathbb{N}$ 是一个整形函数，$S_n^{m(n)-\ell(n)}$ 是哈希族。对于每个 $x\in\{0,1\}^n$ 和 $h\in S_n^{m(n)-\ell(n)}$，定义

$$F(x,h)\stackrel{\text{def}}{=}(f(x),h(x),h)$$

如果可以在多项式时间内计算 f，而且可以在 $\text{poly}(n)$ 时间内通过 n 计算 $m(n)$，那么也能在多项式时间内计算出 F。下面我们证明：如果 f 是正则单向函数，那么对 F 求逆是困难的。而且，若 $l(\cdot)$ 是对数函数，则 F 是"近似 1-1"的。

命题 3.5.9：设 f,m,l,F 如构造 3.5.8 所述。假设存在这样一个算法：输入 n 时，此算法可以在 $\text{poly}(n)$ 时间内计算 $m(n)$。那么：

1. F 是"近似 1-1"的：

$$\Pr[|F^{-1}F(U_n,H_n^{m(n)-\ell(n)})|>2^{\ell(n)+1}]<O(n\cdot 2^{-\ell(n)/4})$$

（H_n^k 表示在 S_n^k 上均匀分布的随机变量。）

2. F"保持"了 f 的单向性：

若 f 是强（或弱）单向的，则 F 亦然。

略证 第一部分可以通过引理 3.5.1，利用 $S_n^{m(n)-l(n)}$ 是哈希族这一前提条件来证明。特别地，引理 3.5.1 隐含了：对每个 α 和几乎 $2^{-l(n)}$ 的 $h \in S_n^{m(n)-l(n)}$，$\Pr[h(U_n)=\alpha] \le 2^{-m(n)+l(n)-1}$ 成立。于是，对所有的 α，有下面这个不等式成立：$\Pr[|F^{-1}(\alpha, H_n^{m(n)-l(n)})| > 2^{l(n)+1}] < 2^{-l(n)}$。根据（后面将给出的）结论 3.5.9.1，可以证明：$\Pr[(H_n^{m(n)-l(n)}(U_n), H_n^{m(n)-l(n)}) \in B] < O(m(n) \cdot 2^{-l(n)})^{1/4}$，这正是第一部分所要求的。

第二部分可以通过可约性理论证明。相反地，假设存在一个能以不允许的成功概率对 f 求逆的有效算法 A，那么我们就可以构造出一个能以不允许的成功概率对 f 求逆的有效算法 A'（从而得到矛盾）。具体地，我们可以考虑如下情况：f 是强单向函数，相反地，假设输入 $F(U_n, H_n^{m(n)-l(n)})$ 时算法 A 能以概率 $\varepsilon(n)$ 对 F 求逆，使得对无穷多的 n 有 $\varepsilon(n) > \frac{1}{\text{poly}(n)}$。以下是对 A' 的描述。

输入 y（设其在 $f(U_n)$ 的值域内）时，算法 A' 均匀地选择 $h \in S_n^{m(n)-l(n)}$ 和 $\alpha \in \{0,1\}^{m(n)-l(n)}$，并用 (y,α,h) 初始化 A。算法 A' 把 x 设为 $A(y,\alpha,h)$ 的 n 位前缀，并输出 x。

显然，算法 A' 在多项式时间内运行。下面我们计算 A' 的成功概率。对算法 A' 每个可能输入的 y，我们考虑一个在 $f^{-1}(y)$ 上均匀分布的随机变量 X_n（即，若 $\alpha \in f^{-1}(y)$，则 $\Pr[X_n=\alpha]=2^{-m(n)}$，否则 $\Pr[X_n=\alpha]=0$）。设 $\delta(y)$ 为输入 $(y, H_n^k(X_n), H_n^k)$ 时算法 A 的成功概率，其中 $n \stackrel{\text{def}}{=} |y|$，$k \stackrel{\text{def}}{=} m(n)-l(n)$。也即，

$$\delta(y) \stackrel{\text{def}}{=} \Pr[A(y, H_n^k(X_n), H_n^k) \in F^{-1}(y, H_n^k(X_n), H_n^k)] \tag{3.14}$$

根据矛盾假设（和 $\delta(y)$ 的定义）可知，$\mathrm{E}[\delta(f(U_n))]=\varepsilon(n)$ 成立，$\Pr[\delta(f(U_n)) > \frac{\varepsilon(n)}{2}] > \frac{\varepsilon(n)}{2}$，得证。任意选定一个 $y \in \{0,1\}^n$，使得 $\delta(y) > \frac{\varepsilon(n)}{2}$，证明如下抽象结论。

结论 3.5.9.1：设 k 为自然数，且 $k \le n$，再令 $X_n \in \{0,1\}^n$ 是一个随机变量，对所有 $x \in \{0,1\}^n$，满足 $\Pr[X_n = x] \le 2^{-k}$。假设 B 是一个二元集合，且

$$\delta \stackrel{\text{def}}{=} \Pr[(H_n^k(X_n), H_n^k) \in B]$$

则

$$\Pr[(U_k, H_n^k) \in B] > \frac{\delta^4}{O(k)}$$

利用 A' 的定义，并把结论 3.5.9.1 作用于等式 3.14，可以证明输入 y 时 A' 能对 f 求逆的概率等于

$$\Pr[A(y, U_k, H_n^k) \in F^{-1}(y, U_k, H_n^k)] > \frac{\delta(y)^4}{O(k)} > \frac{\delta(y)^4}{O(n)} \tag{3.15}$$

于是，

$$\begin{aligned}
&\Pr[A'(f(U_n)) \in f^{-1}(f(U_n))] \\
&\ge \Pr\left[\delta(f(U_n)) > \frac{\varepsilon(n)}{2}\right] \cdot \Pr\left[A'(f(U_n)) \in f^{-1}(f(U_n)) \,\middle|\, \delta(f(U_n)) > \frac{\varepsilon(n)}{2}\right] \\
&> \frac{\varepsilon(n)}{2} \cdot \frac{(\varepsilon(y)/2)^4}{O(n)}
\end{aligned}$$

于是我们推出（与 f 是强单向函数的前提）矛盾，命题得证[①]。剩下的就是要证明论题 3.5.9.1。下面即将给出的这一证明过程是相当抽象的。

需要强调的是，$m(n)$ 可以通过 n 计算出来这一结论在可约性理论中并不是必不可少的（因为也可以尝试 $m(n)$ 所有可能的取值）。

结论 3.5.9.1 本身是有用的。但是它的证明过程并没有什么重要的思想，可以忽略而不会造成什么影响（特别是对那些对密码学比对“概率分析”更感兴趣的读者）。

结论 3.5.9.1 的证明：首先我们利用引理 3.5.1 证明：S_n^k 中只有“小部分”哈希函数可以把一个“大”概率集映射成多个“小”子集。一旦得到证明，就可以按如下步骤来证明此结论：除去很小一部分无效函数，然后在函数不是无效的条件下，关联结论中出现的两个概率。详述如下。

首先我们限定把一个“大”概率集映射成多个“小”子集的哈希函数的比例。如果存在一个基数为 $\Psi \cdot 2^k$ 的集合 $R \subset \{0,1\}^k$，使得 $\Pr[h(X_n) \in R] \geq (T+1) \cdot \Psi$，那么我们就说函数 $h \in S_n^k$ 是 (T,Ψ)-扩展的。也就是说，h 把一个概率集映射到某一（密度为 Ψ 的）小集合上，其中此概率集的密度是小集合的 $T+1$ 倍（即 h 把一个相对较大的概率集映射到这个集合上）。我们的第一个目标就是要证明：对某一常数 $c>0$，最多有 $\frac{\delta}{4}$ 的 h 是 $(\frac{c \cdot k}{\delta^2}, \frac{\delta^3}{3c \cdot k})$-扩展的。换句话说，只有 $\frac{\delta}{4}$ 的函数是把密度大于 $(\frac{c \cdot k}{\delta^2}+1) \cdot \frac{\delta^3}{3c \cdot k} \approx \frac{\delta}{3}$ 的概率集映射到某一密度为 $\frac{\delta^3}{3c \cdot k}$ 的集合上。

下面我们先证明一个相关问题。如果 $\Pr[h(X_n)=\alpha] \geq (t+1) \cdot 2^{-k}$，那么我们就称 $\alpha \in \{0,1\}^k$ 对 h 是 t-超载的。如果存在一个基数为 $\rho 2^k$ 的集合 $R \subset \{0,1\}^k$，使得每个 $\alpha \in R$ 对 h 是 t-超载的，那么我们就称函数 $h \in S_n^k$ 是 (t,ρ)-超载的（显然，如果 h 是 (t,ρ)-超载的，那么它也是 (t,ρ)-扩展的，但是反过来不一定成立）。我们首先证明最多有 $\frac{1}{t^2\rho}$ 的 h 是 (t,ρ)-超载的。证明分两步：

1. 已知对每个 x，有 $\Pr[X_n=x] \leq 2^{-k}$。利用引理 3.5.1，可以证明每个 $\alpha \in \{0,1\}^k$ 最多对 t^{-2} 的 h 来说是 t-超载的。

2. 考虑一个双向图，其一边是 (t,ρ)-超载函数，另一边是一些 k 比特长的字符串，使得 $(h,\alpha) \in S_n^k \times \{0,1\}^k$ 是该图的边当且仅当 α 对 h 是 t-超载的。这个图中，每个 (t,ρ)-超载函数的度至少是 $\rho \cdot 2^k$，而每个 (t,ρ)-超载函数像 α 的度最多是 $t^{-2} \cdot |S_n^k|$。因此，(t,ρ)-超载函数最多有 $\frac{2^k \cdot (t^{-2} \cdot |S_n^k|)}{\rho \cdot 2^k} = \frac{1}{t^2 \rho} \cdot |S_n^k|$ 个。

现在我们就把扩展和超载性相关联，这就能通过超载函数的个数的上界得到扩展函数的个数的上界（这是非平凡方向）。详细证明如下。

---

① 如果 f 是弱单向的，则需将此证明过程稍作修改。特别地，假设对某一正多项式，任一概率多项式时间算法在输入 $f(U_n)$ 时不能对 f 求逆的概率至少是 $1/p(n)$。我们可以断言：任一概率多项式时间算法在输入 $F(U_n, H_n^k)$ 时不能对 F 求逆的概率至少是 $1/4p(n)$。相反地，假设存在一个概率多项式时间算法，在输入 $F(U_n, H_n^k)$ 时能对 F 求逆的概率至少是 $1-\epsilon(n)$，其中 $\epsilon(n) \leq 1/4p(n)$。那么如前所述，对 $\delta(\cdot)$，有 $\mathsf{E}[\delta(f(U_n))]=1-\epsilon(n)$ 和 $\Pr[\delta(f(U_n)) \geq 1-2p(n)\epsilon(n)] \geq 1-\frac{1}{2p(n)}$ 成立。再根据 $\epsilon(n) \leq 1/4p(n)$，我们可以推断：至少对 $1-\frac{1}{2p(n)}$ 的 n 比特长的字符串 x，$\delta(f(x)) \geq \frac{1}{2}$ 成立。利用论题 3.5.9.1 可以证明：对这些 x 来说，输入 $f(x)$ 时 A' 能对 f 求逆的概率是 $\Omega(1/n)$。如果某一算法 $O(n^2)$ 次重复 A'，那么我们就能得到一个概率多项式时间算法，输入 $f(U_n)$ 时，此算法对 f 求逆成功的概率至少是 $(1-\frac{1}{2p(n)}) \cdot (1-2^{-n}) > 1-\frac{1}{p(n)}$，这与我们关于 f 的假设相矛盾。

子结论:对 $T \geq 4$,若 h 是 (T,Ψ)-扩展的,则存在一个整数 $i \in \{1,\cdots,k+2\}$,使得 h 是 $(T \cdot 2^{i-3}, \frac{\Psi}{k \cdot 2^{i+1}})$-超载的。

子结论证明如下:设 R 是基数为 $\Psi \cdot 2^k$ 的集合,满足 $\Pr[h(X_n) \in R] \geq (T+1) \cdot \Psi$。对 $i=1,\ldots,k+3$,令 $R_i \subseteq R$ 表示 R 中各点的子集,对 h 是 $(2^{i-3} \cdot T)$-超载的(当然,$R_{k+3}=\phi$)。相反地,假设对每个 $i \in \{1,\ldots,k+2\}$,有 $|R_i| < \frac{\Psi}{k \cdot 2^{i+1}} \cdot 2^k$,那么对 $T \geq 4$ 和 $k \geq 1$,有

$$\begin{aligned}
\Pr[h(X_n) \in R] &= \Pr[h(X_n) \in (R \setminus R_1)] + \Pr[h(X_n) \in \bigcup_{i=1}^{k+2} (R_i \setminus R_{i+1})] \\
&\leq \left(\frac{T}{4}+1\right) \cdot \Psi + \sum_{i=1}^{k+2} (2^{(i+1)-3} \cdot T + 1) \cdot \frac{|R_i \setminus R_{i+1}|}{2^k} \\
&< \left(\frac{T}{4}+1\right) \cdot \Psi + \sum_{i=1}^{k+2} (2^{i-2} \cdot T + 1) \cdot \frac{\Psi}{k \cdot 2^{i+1}} \\
&\leq (T+1) \cdot \Psi
\end{aligned}$$

这与有关 R 的前提假设矛盾。

由这个(对任意 $T \geq 4$ 和 $\Psi > 0$ 成立的)子结论可得:(T,Ψ)-扩展的 h 所占的比例不超过

$$\sum_{i=1}^{k} \frac{1}{(T \cdot 2^{i-3})^2 \cdot \frac{\Psi}{k \cdot 2^{i+1}}} < \frac{128k}{T^2 \cdot \Psi}$$

其中和式的第 i 项即为 $(T \cdot 2^{i-3}, \frac{\Psi}{k \cdot 2^{i+1}})$-超载的 h 所占比例的上界。对 $c=1536$,令 $T=\frac{ck}{\delta^2}$,$\Psi=\frac{\delta^3}{3ck}$,可得最多有 $\frac{128k}{(ck/\delta^2)^2 \cdot (\delta^3/3ck)}=\frac{\delta}{4}$ 的 h 是 $(\frac{ck}{\delta^2},\frac{\delta^3}{3ck})$-扩展的。

已经得到了适当的扩展函数的上界,现在我们就可以开始证明这个结论了。特别地,如果 h 不是 $(\frac{1536k}{\delta^2},\frac{\delta^4}{4608k})$-扩展的,我们则称 h 是可信的。关于可信函数有两个重要事实:

事实 1:最多只有 $\frac{\delta}{4}$ 的 h 是可信的。

事实 2:若 h 是可信的且 $\Pr[h(X_n) \in R] \geq \frac{\delta}{2}$,则 $\Pr[U_k \in R] \leq \frac{\delta^3}{4608k}$。(假设 h 是可信的且 $\Pr[U_k \in R] \geq \frac{\delta^3}{4608k}$ 成立,那么 $\Pr[h(X_n) \in R] < (\frac{1536k}{\delta^2}+1) \cdot \frac{\delta^3}{4608k} = \frac{\delta}{3} + \frac{\delta^3}{4608k} < \frac{\delta}{2}$。)

针对可信的 h,现在我们计算均匀选择 α 时 (α,h) 和 B 冲突的概率。如果 $\Pr[(h(X_n),h) \in B] \geq \frac{\delta}{2}$,我们就称 h 是好的。根据 Markov 不等式(和 δ 的定义),得到如下事实。

事实 3:H_n^k 是好的的概率至少是 $\frac{\delta}{2}$。

用 P(意为"perfect")表示既是好的又是可信的 h 的集合。结合事实 1 和 3,得到事实 4。

事实 4:$\Pr[H_n^k \in P] \geq \frac{\delta}{2} - \frac{\delta}{4} = \frac{\delta}{4}$。

令 $B_h \stackrel{\text{def}}{=} \{\alpha:(\alpha,h) \in B\}$。显然,对每个 $h \in P$,有 $\Pr[h(X_n) \in B_h] \geq \frac{\delta}{2}$(因为 h 是好的),以及 $\Pr[U_k \in B_h] \geq \frac{\delta^3}{4608k}$(因为 h 是可信的,再把事实 2 这一前提作用于 B_h)。于是:

事实 5:对每个 $h \in P$,$\Pr[(U_k,h) \in B] \geq \frac{\delta^3}{4608k}$ 成立。

结合事实 4 和 5,有

$$\begin{aligned}
\Pr[(U_k,H_n^k) \in B] &\geq \Pr[(U_k,H_n^k) \in B \mid H_n^k \in P] \cdot \Pr[H_n^k \in P] \\
&\geq \frac{\delta^3}{4608k} \cdot \frac{\delta}{4}
\end{aligned}$$

结论得证。

应用命题 3.5.9

把构造 3.5.2 作用于构造 3.5.8 产生的函数是可能的，并且只需稍作修改，命题 3.5.3 仍然成立。特别地，对于每一个正多项式，修改后的边界（与命题 3.5.3 类似）是 $2^{-\Omega(l(n))}+\frac{1}{p(n)}$（而不是 $2\cdot 2^{-\frac{l(n)}{3}}+\frac{1}{p(n)}$）。这个结论对定理 3.5.6 也有效。而且，我们甚至可以放弃 $m(n)$ 是可以计算的这一要求（因为我们可以对每一个可能的 $m(n)$ 构造函数 F_m）。最后，需要注意的是，即使按如下方式放宽对正则函数的定义，整个结论仍然成立。

定义 3.5.10（正则函数，修改后的定义）：如果存在一个整型函数 m'，那么函数 $f:\{0,1\}^*\to\{0,1\}^*$ 被称为正则函数：$\mathbb{N}\to\mathbb{N}$ 和一个多项式 $q(\cdot)$，使得对所有足够长的 $x\in\{0,1\}^*$，下式成立：

$$2^{m'(|x|)}\leq|\{y:f(x)=f(y)\}|\leq q(|x|)\cdot 2^{m'(|x|)}$$

在构造 3.5.8 中运用这些定义更为宽松的正则函数时，令 $m(n)\stackrel{\text{def}}{=}m'(n)$。得到的函数将具有稍弱点的近似 1-1 的性质。亦即，

$$\Pr[|F^{-1}(F(U_n,H_n^{m(n),l(n)}))|>q(n)\cdot 2^{l(n)-1}]<2^{-\Omega(l(n))}$$

构造 3.5.2 的应用也将做相应的修改。可以得到：

定理 3.5.11：若正则单向函数存在，则伪随机发生器也存在。

3.5.3 在正则单向函数之后的讨论

命题 3.5.9 的证明主要依靠单向函数 f（至少从非严格意义上说）是正则的这一事实。或者必须修改构造 3.5.8，使得不同的哈希函数与不同的 $x\in\{0,1\}^n$ 相关联。而且，定理 3.5.6 的证明过程不能重复，除非对给定的 x，集合 $f^{-1}(f(x))$ 的基数是易于计算的。需要注意的是，此时我们不能为每个可能的 $\lceil\log_2|f^{-1}(y)|\rceil$ 构造函数 F_m，因为这些函数都不满足命题 3.5.9。于是又需要一种新的思路。

一个关键理论是：尽管 $\log_2|f^{-1}(f(x))|$ 的值对不同的 $x\in\{0,1\}^n$ 来说是可变的，但是 $m(n)\stackrel{\text{def}}{=}\mathsf{E}(\log_2|f^{-1}(f(U_n))|)$ 的值是惟一的。若函数 $\overline{f}$ 定义为：

$$\overline{f}(x_1,\ldots,x_{n^2})\stackrel{\text{def}}{=}f(x_1),\ldots,f(x_{n^2})$$

其中，$|x_1|=\cdots=|x_{n^2}|=n$，则它具有如下性质：这个域上属于原像集合的部分几乎是可忽略不计的，而且它们的基数的对数并没有偏离期望值太多。特别地，若令 $\overline{m}(n^3)\stackrel{\text{def}}{=}\mathsf{E}(\log_2|\overline{f}^{-1}(\overline{f}(U_{n^3}))|)$，显然有 $\overline{m}(n^3)=n^2\cdot m(n)$。利用 Chernoff 边界，可得

$$\Pr[|\overline{m}(n^3)-\log_2|\overline{f}^{-1}(\overline{f}(U_{n^3}))||>n^2]<2^{-n}$$

假设我们把构造 3.5.8 作用于 $\overline{f}$，令 $l(n^3)\stackrel{\text{def}}{=}n^2$，得到的函数记为 $\overline{F}$。再把构造 3.5.2 作用于 $\overline{F}$，此时令 $l(n^3)\stackrel{\text{def}}{=}2n^2-1$。利用命题 3.5.3 和 3.5.9 的证明过程中介绍的思想，我们可以证明，如果构造 3.5.2 中所使用的把 n^3 比特映射为 $l(n^3)+1$ 比特的函数是 $\overline{F}$ 的核心函数，那么得到的算法就是一个伪随机发生器。这样剩下的问题就是为函数 $\overline{F}$ 构造一个极大核心函数 $\overline{G}$。特别地，对所有 x 来说，$|\overline{G}(x)|$ 都必须等于 $2|x|^{\frac{2}{3}}$。一个很自然的想法就是和构造 3.5.4 中定义 $\overline{g}$ 类似地定义 $\overline{G}$。不幸的是，我们并不知道如何证明这个构造（作用于 $\overline{F}$ 时）的有效性，因此需要一个更复杂的构造。这个构造所利用的方法和前面的方法不完全相同，还包括一些本书未提及的附加方法。这个构造的有效性的证明过程更加复杂，不适合在本书中讨

论。因此我们只给出这个结论。

定理 3.5.12：若单向函数存在，则伪随机发生器也存在。

随便说明一下(并以此作为总结)，定理 3.5.12 的非均匀复杂度对等体也是有效的，事实上它是很容易确定的：

定理 3.5.13：假设(如定义 2.2.6 所述的)非均匀单向函数存在，那么伪随机发生器也存在。而且，利用(如定义 3.2.7 所述的)多项式大小的电路是不能把这些发生器的输出总体同均匀总体区别开的。

3.6 伪随机函数

本节我们介绍伪随机函数的定义和构造方法(利用任意的伪随机发生器作为一个模块)。伪随机函数在第 2 册的第 5 章和第 6 章都起着重要的工具作用。

动机 伪随机发生器使我们能够利用非常少数的伪随机位，来产生、交换和共享很大数目的伪随机值。特别地，可以利用 n 个(均匀选择的)随机位产生、交换并共享 poly(n)个伪随机位。因为任意有效的应用程序仅使用多项式多的随机值，从而能够访问多项式多的伪随机入口(*entries*)对这样的应用程序已经足够。但这个结论太草率了，因为它隐含地假设了这些入口(即访问地址)是预先固定的。在一些正常的应用程序里，可能需要访问由应用程序"动态地"确定的地址。例如，我们可能需要赋值给在整个程序运行过程中产生的(多项式)多的 n 位长的串，以使这些值在后面能重新恢复。利用伪随机发生器，只要产生 n 个随机位，并存储 poly(n)多的值，就可以达到上述目的。本节的挑战是仅产生 n 个随机位，仅存储 n 个随机位来达到上述目的。解决问题的关键是伪随机函数的概念。直观地，一组用户共享的一个伪随机函数对于非法用户来说(工作组以外的)是一个随机函数。

3.6.1 定义

一般地说，伪随机函数就是，任意有效的能够由它选择的自变量的值计算函数值的算法都不能将它与真随机函数区分开来的函数。因此，分辨程序可能在不同的点对被检函数质疑，可能依靠前面得到的答案，但仍然不能分辨答案是由来自伪随机函数总体的函数提供的，还是由来自均匀函数总体的函数提供的。实际上，为了把伪随机函数的概念正式化，我们需要考虑函数总体。为了简单起见，我们可以考虑等长度的函数总体，在后面我们鼓励读者可以通过令 $l(n)=n$ 进一步简化我们的讨论。概括性的讨论将在 3.6.4 小节进行。

定义 3.6.1(函数总体)：设 $l:\mathbb{N}\to\mathbb{N}$ (例如，$l(n)=n$)。一个 l 位的函数总体是一个随机变量序列 $F=\{F_n\}_{n\in\mathbb{N}}$，使得随机变量 F_n 在一个从 $l(n)$位长的串映射到 $l(n)$位长的串的函数集合内取值。均匀的 l 位的函数总体记为 $H=\{H_n\}_{n\in\mathbb{N}}$，其中 H_n 服从所有从 $l(n)$位长的串映射到 $l(n)$位长的串的函数集合上的均匀分布。

为了将伪随机函数的概念正式化，我们利用(概率多项式时间)预言机(见 1.3.5 小节)。我们这里使用术语"预言机"与标准用法几乎是等同的。一个细微的差别就是，我们考虑的预言机以等长度的函数而不是布尔函数作为预言(而后者在复杂性理论中更标准)。此外，我们

假设当输入 1^n 时，预言机只对长度 $l(n)$ 质疑。这些约定并不是必要的（他们仅仅使我们的说明简化了一点点）。我们记预言机 M 访问预言 f 时的运行过程为 M^f。

定义 3.6.2（伪随机函数总体）：一个 l 位的函数总体 $F=\{F_n\}_{n\in\mathbb{N}}$ 称为是伪随机的，如果对每个概率多项式时间预言机 M、每一个多项式 $p(\cdot)$ 和所有足够大的 n，都有

$$|\Pr[M^{F_n}(1^n)=1]-\Pr[M^{H_n}(1^n)=1]|<\frac{1}{p(n)}$$

其中 $H=\{H_n\}_{n\in\mathbb{N}}$ 是均匀的 l 位的函数总体。

利用与命题 3.2.3（在 3.2.2 小节）的证明中相似的技巧，我们能说明存在与均匀总体不统计接近的伪随机总体。然而，为了实际应用，我们要求伪随机函数可以有效计算。也就是说，函数总体必须有简洁的代表，既能选择它们也能评估它们。下面的定义满足这一点，其中 I 是一个选择函数代表的算法（通过映射 ϕ 与函数自身相关联）。

定义 3.6.3（可有效计算的函数总体）：一个 l 位的函数总体 $F=\{F_n\}_{n\in\mathbb{N}}$，如果满足下面两个条件：

1. 有效索引（efficient indexing）：存在一个概率多项式时间算法 I，一个由串到函数的映射 ϕ，使得 $\phi(I(1^n))$ 和 F_n 服从相同的分布。

我们记对应于串 i 的函数为 f_i（即，$f_i\stackrel{\text{def}}{=}\phi(i)$）。

2. 有效评估：存在一个多项式时间算法 V，使得对于每一个 $I(1^n)$ 中的 i 和 $x\in\{0,1\}^{l(n)}$，都有 $V(i,x)=f_i(x)$。

则称它为可有效计算的总体。

特别地，在可有效计算的总体中的函数都有相对简洁的表示法（即，有 n 的多项式而不是以 n 为指数的长度）。从而，可有效计算的函数总体只能有指数多的函数（假设 $l(n)=n$，会有超过双指数多的函数）。

值得强调的另一点是，可有效计算的伪随机函数只要相应的描述也给定，就可以被有效评估。然而，如果函数（或它的描述）不知道，在给定点处函数的值就不可估计，即使在非常宽松的意义下，甚至给定函数在其他点处的值，上述值仍然是不可估计的。

术语　本书的剩余部分只考虑可有效计算的伪随机函数总体。因此，当我们讨论伪随机函数时，我们实际指的是从可有效计算的伪随机函数总体中随机选择的函数。

不失一般性，观察在定义 3.6.3 中的索引算法中用到的掷币序列可以作为函数的描述。将这个观察结果与定义 3.6.2 相结合，我们得到下面的可有效计算的伪随机函数的候选定义。

定义 3.6.4（可有效计算的伪随机函数总体，候选形式）：一个可有效计算的伪随机函数总体（伪随机函数）是一个有限函数集合

$$\{f_s:\{0,1\}^{l(|s|)}\to\{0,1\}^{l(|s|)}\}_{s\in\{0,1\}^*}$$

这里，$l:\mathbb{N}\to\mathbb{N}$，且下面的两个条件成立：

1. 有效估计：存在一个多项式时间算法，当输入 s 和 $x\in\{0,1\}^{l(|s|)}$ 时，输出 $f_s(x)$。

2. 伪随机性：如果函数总体 $F=\{F_n\}_{n\in\mathbb{N}}$，通过定义使得 F_n 服从多重集 $\{f_s\}_{s\in\{0,1\}^n}$ 上的均匀分布，则总体 $F=\{F_n\}_{n\in\mathbb{N}}$ 是伪随机的。

我们指出可以类似地定义并构造更一般的伪随机函数；见 3.6.4 小节。

3.6.2 构造

利用任何一个伪随机发生器，我们可以构造可有效计算的伪随机总体($l(n)=n$)。

构造 3.6.5：设 G 是一个确定性算法，能够将长为 n 的输入扩充为长为 $2n$ 的串。我们记 $G(s)$ 的 $|s|$ 位长的前缀为 $G_0(s)$，$G(s)$ 的 $|s|$ 位长的后缀为 $G_1(s)$（即，$G(s)=G_0(s)G_1(s)$）。对每一个 $s\in\{0,1\}^n$，我们定义一个函数 $f_s:\{0,1\}^n\to\{0,1\}^n$，使得对每一个 $\sigma_1,\ldots\sigma_n\in\{0,1\}$，都有

$$f_s(\sigma_1\sigma_2\cdots\sigma_n)\stackrel{\text{def}}{=}G_{\sigma_n}(\cdots(G_{\sigma_2}(G_{\sigma_1}(s)))\cdots)$$

也就是，当输入 s 和 $x=\sigma_1\sigma_2\cdots\sigma_n$ 时，如下计算值 $f_s(x)$：

```
Let y = s.
For i = 1 to n, do
y ← G_σi(y).
Output y.
```

设 F_n 是通过均匀选择 $s\in\{0,1\}^n$，并令 $F_n=f_s$ 而定义的随机变量。最后令 $F=\{F_n\}_{n\in\mathbb{N}}$是我们的函数总体。

图解见图 3.5，函数 f_s 通过一个有节点标记的深度为 n 的完全二叉树的 n 步移动来定义。树的根节点，后面指树的 0 层节点，标记为串 s。如果一个内部节点标有 r，则它的左孩子标记为 $G_0(r)$，而它的右孩子标记为 $G_1(r)$。$f_s(x)$的值为位于叶子上的由根节点通过与串 x 相应的路径可到达的串。随机变量 F_n 的取值为这些已编号的树，这些树是与相应的所有 2^n 种可能的根节点的编号所对应的树，且 F_n 服从均匀分布。

上面构造的作用于 n 位的串的函数完全由这 n 个位所决定。因此，选择、交换和存储这样的函数可以通过选择、交换和存储一个 n 位的串来实现。

定理 3.6.6：设 G 和 F 和构造 3.6.5 中的一样，并假设 G 是一个伪随机发生器，那么 F 是一个可有效计算的伪随机函数总体。

结合定理 3.5.12 和定理 3.6.6，我们可以得到如下推论。

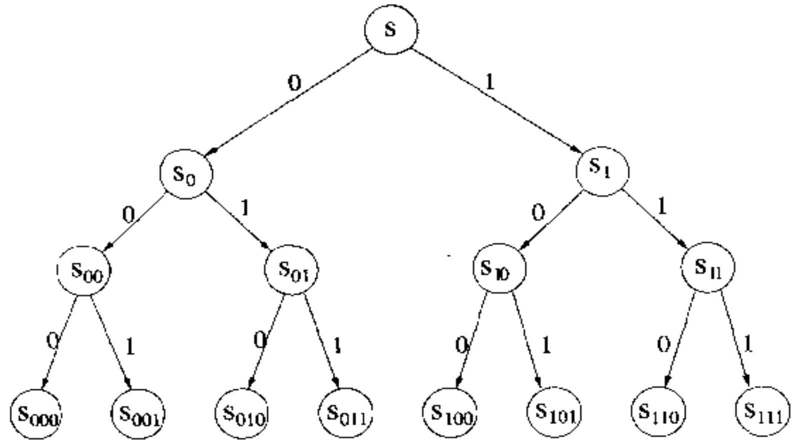
推论 3.6.7：如果单向函数存在，则伪随机函数也存在。

同理，结合定理 3.6.6 和 $l(n)>\log_2 n$ 时的情形，任意伪随机函数（和定义 3.6.4 中一样）都可以产生伪随机发生器（见习题 28），我们得到如下推论。

推论 3.6.8：伪随机函数（对于超对数）存在当且仅当伪随机发生器存在。

定理 3.6.6 的证明：显然，总体明 F 是可有效计算的。为证明 F 是伪随机的，我们使用混合技术。我们给第 k 个混合赋值这样的函数，它由为树的第 k 层（最高层）节点均匀选择的标号决定，并和构造 3.6.5 中一样计算较低层节点的标号。0 混合与随机变量 F_n 相对应（因为给根节点赋了一个随机选择的标号），而第 n 个混合对应于均匀随机变量 H_n（因为每个叶子都有一个均匀选择的标号）。下面我们说明一个分辨相邻混合的有效预言机能转化为一个算法，此算法能分辨 $G(U_n)$ 的多项式多的样本和 U_{2n} 的多项式多的样本。利用定理 3.2.6，我们就得出一个与假设（G 是一个伪随机发生器）相悖的矛盾。具体如下。

我们令 $s_\lambda = s$，$s_{\alpha\sigma} = G_\sigma(s_\alpha)$. 则由根节点 S 出发经由路径 $\sigma_1\sigma_2\cdots\sigma_n$. 到达叶子节点的值为 $f_s(\sigma_1\sigma_2\cdots\sigma_n) = s_{\sigma_1\sigma_2\cdots\sigma_n}$



例如，$f_s(001) = s_{001} = G_1(s_{00}) = G_1(G_0(s_0)) = G_1(G_0(G_0(s)))$.

图 3.5

对每一个 k，满足 $0 \leq k \leq n$，我们定义一个混合分布 H_n^k，如下取值为这样的函数 $f:\{0,1\}^n \to \{0,1\}^n$。对每一个 $s_1, s_2, ..., s_{2^k} \in \{0,1\}^n$，我们定义函数 $f_{s_1,\dots,s_{2^k}}:\{0,1\}^n \to \{0,1\}^n$，使得

$$f_{s_1,\dots,s_{2^k}}(\sigma_1\sigma_2\cdots\sigma_n) \stackrel{\text{def}}{=} G_{\sigma_n}(\cdots(G_{\sigma_{k+2}}(G_{\sigma_{k+1}}(s_{\text{idx}(\sigma_k\cdots\sigma_1)})))\cdots)$$

其中 $\text{idx}(\alpha)$ 是在所有长为 $|\alpha|$ 的二进制串中 α 在标准字典序中的索引号。即，这样来计算 $f_{s_1,\dots,s_{2^k}}(x)$，首先用 x 的 k 位长的前缀确定某一 s_j，然后用 $(n-k)$ 位长的后缀来确定在每一个剩余阶段到底用 G_0 和 G_1 中的哪一个函数（见构造 3.6.5）。随机变量 H_n^k 服从所有可能的 $(2^n)^{2^k}$ 个函数（与所有可能的选择 $s_1, s_2, ..., s_{2^k} \in \{0,1\}^n$ 相对应）上的均匀分布。即，

$$H_n^k \stackrel{\text{def}}{=} f_{U_n^{(1)},\dots,U_n^{(2^k)}}$$

其中每一个 $U_n^{(j)}$ 都服从 $\{0,1\}^n$ 上的均匀分布，且它们相互独立。

显然，H_n^0 与 F_n 是同一个随机变量，H_n^n 与 H_n 是同一个随机变量。再者，像混合技术中经常使用的，能够分辨端点混合就能够分辨一对相邻混合。这种能力进一步变形，就得出与 G 的伪随机性矛盾的结论。具体如下，

与定理相反，我们假设函数总体 F 不是伪随机的。从而存在概率多项式时间预言机 M 和多项式 $p(\cdot)$，使得对于无限多的 n，有

$$\Delta(n) \stackrel{\text{def}}{=} |\Pr[M^{F_n}(1^n) = 1] - \Pr[M^{H_n}(1^n) = 1]| > \frac{1}{p(n)}$$

设 $t(\cdot)$ 是 $M(1^n)$ 的运行时间的一个多项式界（因为 M 是一个多项式时间预言机，所以这样的多项式肯定存在）。从而，当输入 1^n 时，预言机 M 至多进行 $t(n)$ 次质疑（因为，显然质疑次数以运行时间为界）。利用预言机 M，我们可以构造算法 D，它可以分辨总体 $\{G(G_n)\}_{n\in\mathbb{N}}$ 的 $t(n)$-乘积和总体 $\{U_{2n}\}_{n\in\mathbb{N}}$ 的 $t(n)$-乘积，具体如下。

算法 D：当输入 $\alpha_1, ..., \alpha_t \in \{0,1\}^{2n}$（其中，$t = t(n)$）时，算法 D 如下运行。首先 D 均匀地

选择 $k\in\{0,1,...,n-1\}$。这个随机选择在后面称为检测点，是 D 自身做的惟一的随机选择。接着，算法 D 调用预言机 M（输入为 1^n 时的 M）并回答 M 的质疑。对预言机 M 的第一个质疑，记为 q_1，回答为

$$G_{\sigma_n}(\cdots(G_{\sigma_{k+2}}(P_{\sigma_{k+1}}(\alpha_1)))\cdots)$$

其中 $q_1=\sigma_1\cdots,\sigma_n$（$\alpha_1$ 是第一个输入串），$P_0(\alpha)$（$P_1(\alpha)$）表示 α 的 n 位前缀（α 的 n 位后缀）。此外，算法 D 记录这个质疑（即 q_1）。后面的每一个质疑都通过查它的 k 位长的前缀是否等于前面某一个质疑的 k 位长的前缀来回答。当当前质疑的 k 位长的前缀（记为 q_i）不等于前面所有质疑的 k 位长的前缀时，我们将这个前缀与一个新的输入串（即 α_i）相关联。即我们用

$$G_{\sigma_n}(\cdots(G_{\sigma_{k+2}}(P_{\sigma_{k+1}}(\alpha_i)))\cdots)$$

来回答质疑 q_i，其中 $q_i=\sigma_1\cdots\sigma_n$。此外，算法 D 记录当前质疑（即 q_i）。另一种可能的情况是，第 i 个质疑的 k 位长的前缀等于前面某一个质疑的 k 位长的前缀。设 j 为所有 k 位长的前缀等于第 i 个质疑的 k 位长的前缀的质疑的最小序数。（由假设 $j<i$），然后我们记录当前质疑（即 q_i），但用与质疑 q_i 相关联的串（即输入串 α_j）来回答。也即，我们用

$$G_{\sigma_n}(\cdots(G_{\sigma_{k+2}}(P_{\sigma_{k+1}}(\alpha_j)))\cdots)$$

来回答 q_i，其中 $q_i=\sigma_1\cdots\sigma_n$。最后，当预言机 M 终止时，算法 D 以与 M 相同的输出终止。

由图解，算法 D 首先将 α_1 的前后两部分放到相应的树的孩子节点中，也就是由根节点经过相应的路径 $\sigma_1\cdots\sigma_k$ 能够到达的孩子节点，从而回答第一个质疑。与 $\sigma_1\cdots\sigma_k$ 相应的子树的所有节点的标号都由这两个孩子的标号决定（如 F 的构造中所述）。后面的质疑都通过从根节点出发的相应路径来回答。当此路径不经过已标号的 $(k+1)$ 层节点时，我们给这个节点和它的兄弟赋值一个新的串（输入串）。为了简单起见，当第 i 个质疑的路径要求一个新串时，我们将第 i 个输入串（而不是迄今未用的第一个输入串）赋给它。当一个新质疑的路径经过已标号的某一 $k+1$ 层节点时，我们用这个标号来计算此路径上后面节点的标号（特别是叶子节点的标号）。我们必须指出，此算法不能计算与 $\sigma_1\cdots\sigma_k$ 相应的子树的所有节点的标号（尽管这些标号由与 $\sigma_1\cdots\sigma_k$ 相应的节点的标号决定），只能计算沿此路径的与质疑相应的节点的标号。

显然，算法 D 是多项式时间内可实现的。下面来评估它的性能，关键是观察 D 作用于检测点 k 以及混合 k 和 $k+1$ 上的行为之间的关系。

- 当输入是来自 U_{2n} 的 $t(n)$-乘积时（且算法 D 选择 k 作为检测点），被调用的预言机 M 的行为与作用于第 $k+1$ 个混合时的运行一样。因为 D 将 $2n$ 位长的真随机串赋给了第 $k+1$ 层（这与将 n 位长的真随机串赋给第 $k+1$ 层是一样的）。
- 另一方面，当输入是来自 $G(U_n)$ 的 $t(n)$-乘积时（且算法 D 选择 k 作为检测点），则 M 的行为与作用于第 k 个混合时的运行一样。实际上，D 没有将（未知的）相应的（生成这些伪随机串的）种子赋给第 k 层；但将这些伪随机串的前后两部分赋给第 $k+1$ 层有相同的效果。

因此：

结论 3.6.6.1：设 n 为一个整数，且 $t\stackrel{\text{def}}{=}t(n)$。$K$ 为一个描述（当输入为 t 个 $2n$ 位长的串的序列时）算法 D 随机选择的检测点的随机变量。则对每一个 $k\in\{0,1,...,n-1\}$，

$$\Pr[D(G(U_n^{(1)}),...,G(U_n^{(t)}))=1\mid K=k]=\Pr[M^{H_n^k}(1^n)=1]$$

$$\Pr[D(U_{2n}^{(1)},...,U_{2n}^{(t)})=1\mid K=k]=\Pr[M^{H_n^{k+1}}(1^n)=1]$$

其中每一个 $U_n^{(i)}$ 和 $U_{2n}^{(j)}$ 是相互独立的随机变量，分别服从 $\{0,1\}^n$ 和 $\{0,1\}^{2n}$ 上的均匀分布。

结论 3.6.6.1 是很明显的；然而严格的证明却比直觉意识到的复杂得多，原因是 M 的质疑可能依赖于它前面接收到的答案，从而 D 的输入和赋给混合的可能值之间的关系就不像看起来那么明显。为了说明这个困难，我们考虑在 N 步迭代中运行的一对交互程序选择的 N 位长的串。在每一步迭代中，首先基于交互程序的全部过程选择一个新的定位（即未用过的 $i\in\{1,...,N\}$），接着通过无偏掷币为这个位（即第 i 位）赋值。直观地，结果串服从均匀分布；但我们仍需一个证明（因为随机过程是复杂实体，经常会导致错误）。在我们的讨论中，常常会出现这种情况。确定串的程序在 $T<N$ 步迭代后终止，从而做了关于部分确定的结果串的陈述。因此，这种情况就有点困扰人，必须要有详细的证明。然而，证明没有提供附加的理解，从而可以在没有特别损失的条件下略去不证（特别是对于那些对密码学比对"概率分析"感兴趣的读者）。

结论 3.6.6.1 的证明：我们先在一个特别简单的情况下，即 M 的质疑为前 t 个按字典顺序排列的（长为 n 的）串，草拟论题的一个证明。为了力求简单，我们进一步假设，当输入 $\alpha_1,...,\alpha_t$ 时，算法 D 恰巧选择检测点 k，满足 $t=2^{k+1}$。在这种情况下，当输入为 1^n 时调用预言机 M，并访问函数 $f_{s_1,...,s_{2^{k+1}}}$，其中对每一个 $j\leq 2^k$ 和 $\sigma\in\{0,1\}$ 都有 $s_{2j-1-\sigma}=P_\sigma(\alpha_j)$。因此，如果 D 的输入是在 $\{0,1\}^{2n}$ 中均匀选择的，则访问第 $k+1$ 个混合随机变量时调用 M（因为在这种情况下，每一个 s_j 都相互独立并服从 $\{0,1\}^n$ 上的均匀分布）。另一方面，如果 D 的输入服从的分布与 $G(U_n)$ 相同，则访问第 k 个混合随机变量时调用 M（因为在这种情况下，$f_{s_1,...,s_{2^{k+1}}}=f_{r_1,...,r_{2^k}}$，其中 r_j 是与 α_j 相对应的种子）。

在一般情况下，对每一个 $0\leq m\leq n$，我们考虑另一种候选的方法来定义随机变量 H_n^m。这种候选的方法与 D 回答预言机 M 的质疑的方法有点类似（我们用符号 m 代替 k，因为 m 不必等于算法 D 选择的检测点（记为 k））。这种定义 H_n^m 的方法包含两个相互交错的随机过程，以及最初的随机选择，后而用来确定函数 $g:\{0,1\}^m\to\{0,1\}^n$ 的函数 $g:\{0,1\}^n\to\{0,1\}^n$。第一个随机过程记为 ρ（"外部给定的"），是一个在 g 的定义域内指定点的任意随机过程（过程 ρ 与 M 的质疑相对应，而第二个过程与 A 回答这些质疑的方法相对应）。第二个过程记为 ψ，为 ρ 指定的每一个新点分配均匀选择的 n 位长的串，从而定义了这一点 g 的值。我们指出，当指定一个旧的点时（即，g 在这一点的值已经定义），第二个过程什么也不做（即，g 在这一点的值保持不变）。过程 ρ 可能依赖于这两个过程的历史，特别是为前面的点选择的值。当终止 ρ 时，第二个过程 ψ 为还未定义的点选择随机的值。我们指出 ψ 对于过程 ρ 的所有可能的选择是固定的。本段剩余部分对两个随机过程的交互作用做了详细的描述（可以略去）。我们考虑将 n 位的串序列（代表历史）映射为 m 位串的随机过程 ρ。ρ 不必是无记忆的（从而可能记住它前面的随机选择）。也即，对每一个固定的序列 $v_1,...,v_i\in\{0,1\}^n$，随机变量 $\rho(v_1,...,v_i)$ 服从 $\{0,1\}^m\cup\{\perp\}$ 上的（任意）分布，这里 $\perp$ 是表示终端的一个特殊符号。我们通过如下定义的随机过程 ψ 重复过程 ρ 来定义"随机"函数。过程 ψ 以对定义域内每个点都未定义的函数 g 为始点。在第 i 次迭代过程中，ψ 令 $p_i\overset{\text{def}}{=}\rho(v_1,...,v_{i-1})$，并假设 $p_i\neq\perp$，如果对一些 $j<i$，$p_i=p_j$，则令 $v_i\overset{\text{def}}{=}v_j$，否则令 v_i 服从 F_n 上的均匀分布。在后一种情况下（即，p_i 是新的，从而 g 在 p_i 上没有定义），ψ 令 $g(p_i)\overset{\text{def}}{=}v_i$（事实上，对一些 $j<i$，当 $p_i=p_j$ 时，也有 $g(p_i)=g(p_j)=v_j=v_i$）。当 ρ 终止时（即对一些 T，$\rho(v_1,...,v_T)=\perp$），在 g 没有定义的点上，过程 ψ 在 $\{0,1\}^n$ 内独立均匀地为 g 选择值（如果有必要），从而完成对 g 的定义（或者，我们可以扩充过程 g，使它

在指定所有可能的 m 位的串后才终止)。

一旦一个函数 $g:\{0,1\}^m \to \{0,1\}^n$ 完全定义,我们就可以定义函数 $f^g:\{0,1\}^n \to \{0,1\}^n$

$$f^g(\sigma_1\sigma_2\cdots\sigma_n) \stackrel{\text{def}}{=} G_{\sigma_n}(\cdots(G_{\sigma_{m+2}}(G_{\sigma_{m+1}}(g(\sigma_m\cdots\sigma_1))))\cdots)$$

读者很容易证明 $f^g = f_{g(0^m),\cdots,g(1^m)}$(如前面混合构造中所定义的)。同时,我们很容易证明前面的随机过程(即 ψ 与任意的 ρ 交错作用的过程)能够生成一个函数 g,g 服从由 m 位的串映射为 n 位的串的所有可能的函数集合上的均匀分布。从而前面描述的随机过程就生成这样一个与随机变量 H_n^m 服从相同分布的函数。

现在假设 D 选择的检测点等于 k,且 D 的输入是在 $\{0,1\}^{2n}$ 内均匀选择的。在这种情况下,D 回答 M 的质疑的方式可以看作是将独立均匀选择的 n 位的串作为标号赋给 $k+1$ 层的节点。从而 D 回答 M 质疑的方法就与前面描述的 $m=k+1$ 的过程相一致(其中 M 起着 ρ 的作用,A 起着 ψ 的作用)。因此,在这种情况下,以访问第 $k+1$ 个混合随机变量来调用 M。

另一方面,假设 D 的输入是相互独立的(D 选择的检测点仍等于 k),且每一个都与 $G(U_n)$ 服从相同的分布。在这种情况下,D 回答 M 质疑的方式可以看作是将独立均匀选择的 n 位的串作为标号赋给第 k 层的节点。从而,D 回答 M 的质疑的方法就与前面描述的 $m=k$ 的过程相一致。因此,在这种情况下,以访问第 k 个混合随机变量来调用 M。

结合结论 3.6.6.1 和 $\Delta(n)=\Pr[M^{H_n^0}(1^n)=1]-\Pr[M^{H_n^n}(1^n)=1]$,我们有

$$\begin{aligned}&\Pr[D(G(U_n^{(1)}),\ldots,G(U_n^{(t)}))=1]-\Pr[D(U_{2n}^{(1)},\ldots,U_{2n}^{(t)})=1]\\&=\left(\frac{1}{n}\sum_{k=0}^{n-1}\Pr[M^{H_n^k}(1^n)=1]\right)-\left(\frac{1}{n}\sum_{k=0}^{n-1}\Pr[M^{H_n^{k+1}}(1^n)=1]\right)\\&=\frac{\Delta(n)}{n}\end{aligned}$$

由我们的矛盾假设,此概率对无限多的 n,都比 $\frac{1}{n\cdot p(n)}$ 大。从而 D(一个概率多项式时间算法)就能分辨 $G(U_n)$ 的多项式多的样本与 U_{2n} 的多项式多的样本。利用定理 3.2.6,我们得到与前提(当前定理的)即 G 是一个伪随机发生器相矛盾的结论,从而定理得证。

3.6.3 应用程序:一个一般的方法论

共享伪随机函数使得共享成员可以依据对环境(由推理是不必知道的)的现有看法来确定看似随机的值。为了正确评价这个工具的优劣性,我们必须意识到共享伪随机函数在本质上等同于:当给定的值取自一个很大的可能值的集合时,可以确定随机值与该值相关联。我们强调不需任何通信和同步,就可达成上述协议:无论何时,只要某一成员需要将一个随机值与一个既定值 $v\in\{0,1\}^n$ 相关联,它就会关联一个相同的随机值 $r_v\in\{0,1\}^n$ 给 v。

作为一个说明性的例子,我们考虑辨识朋友或敌人这样一个问题,在这个问题中,一个工作组的成员共享一些秘密,我们希望他们能够互相辨别身份。可能的解决方案是,对于共享秘密的成员共享一个定义域很大的秘密函数,通过回答给定他们的随机口令,用秘密函数值来验证他们的口令,从而鉴别他们的身份。我们声明,利用伪随机函数作为秘密函数,可以保证攻击者不能伪装为成员,即使攻击者逼迫成员回答他们的口令,经过多项式多次的交互作用,他仍然不能伪装。为了证明这个声明,考虑秘密函数是真正随机函数的情况(这仅仅是一个智力测验,因为共享如此大的一个随机对象是不可行的)。在这种情况下,在新点处随机函数的值

（对应于攻击者可能回答的新口令）与其他任意点的值（对应于攻击者逼迫合法用户得到的口令）都是无关的。因此，在这种假想的情况下，攻击者不可能成功。从而在实际情况下（秘密函数是从伪随机总体中选择的），攻击者也注定要失败，否则我们就得到一个能分辨伪随机函数与真随机函数的分辨器。

一般情况下，下面的二步方法论在许多情况下都是有用的：

1. 假设所有的合法用户共享一个随机函数 $f:\{0,1\}^n\to\{0,1\}^n$，来设计我们的算法（攻击者可能从合法用户那里得到验证他们选择的函数 f 的值，但不可能直接访问 f）。① 这一步在证明算法的安全性时达到高潮，假设 f 确实是在所有可能的这样的函数中选择的，而忽略怎样选择并处理这样的一个函数 f 的问题。

2. 用伪随机函数代替随机函数构造一个真正的算法。即，合法用户共享一个随机的/秘密的特定的伪随机函数的种子，而攻击者不知道种子。如前所述，攻击者至多能得到（从合法用户那里）由自己的输入选择所决定的函数值。最后得出结论，这个真实算法（这里给定的）是安全的（否则我们就可以分辨伪随机函数和真随机函数了）。

我们必须指出，这个方法只能应用于合法用户可以共享随机/秘密信息，而攻击者不知道这个信息的情况（例如，在私钥加密算法中）。②

3.6.4* 一般化（普遍化）

这里我们将伪随机函数的概念一般化，首先考虑函数不是等长度的情形，然后考虑定义在所有串集合上的函数的情形。这些推广为应用伪随机函数提供了更大的灵活性。

3.6.4.1 不等长度的函数

从定义 3.6.4 出发，我们介绍下面的伪随机总体概念的一般化。

定义 3.6.9（一般的伪随机函数总体）：设 $d,r:\mathbb{N}\to\mathbb{N}$。如果满足下面的两个条件：

1. 有效估计：存在一个多项式时间算法，当输入 s 和 $x\in\{0,1\}^{d(|s|)}$ 时，返回 $f_s(x)$。

2. 伪随机性：对每一个概率多项式预言机 M、每一个多项式 $p(\cdot)$ 和所有充分大的 n，都有

$$|\Pr[M^{F_n}(1^n)=1]-\Pr[M^{H_n}(1^n)=1]|<\frac{1}{p(n)}$$

其中 F_n 服从多重集 $\{f_s\}_{s\in\{0,1\}^n}$ 上的均匀分布，H_n 服从所有由 $d(n)$ 位长的串映射到 $r(n)$ 位长的串上的均匀分布。

则，我们称

$$\{f_s:\{0,1\}^{d(|s|)}\to\{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$$

是可有效计算的一般的伪随机函数总体（一般的伪随机函数）。

---

① 这与随机预言模型不同，在那里攻击者可以直接访问随机预言（在后面由一个函数来“实现”这个预言，且此函数的描述攻击者也知道）。

② 相比之下，随机预言方法论针对的是这样一种情况，即攻击者也知道函数的描述，这就可以代替攻击者可直接访问的随机预言（如上面的脚注中所说）。我们提醒读者，与这里的算法相比，随机预言方法论是启发性的。进一步讨论见 3.8.2 小节。

显然，$r:\mathbb{N}\to\mathbb{N}$ 必须有一个多项式上界。在定义 3.6.4 中，我们令 d 和 r 都等于函数 l，实际上是定义 3.6.9 的一个特例。与构造 3.6.5 类似，对任意 $d,r:\mathbb{N}\to\mathbb{N}$，其中 $r(n)$ 是在多项式时间 poly(n)内由 n 可计算的，利用任意的伪随机发生器，我们可以构造一般的伪随机函数。特别地：

构造 3.6.10：设 G,G_0,G_1 和构造 3.6.5 中一样。$d,r:\mathbb{N}\to\mathbb{N}$，且设 G'是一个从 n 位长的输入映射到 $r(n)$位长的输出的确定性算法。则对每一个 $s\in\{0,1\}^n$，我们可以定义一个函数 $f_s:\{0,1\}^{d(n)}\to\{0,1\}^{r(n)}$，使得对每一个 $\sigma_1,\ldots,\sigma_{d(n)}\in\{0,1\}$，都有

$$f_s(\sigma_1\sigma_2\cdots\sigma_{d(n)})\stackrel{\text{def}}{=}G'(G_{\sigma_{d(n)}}(\cdots(G_{\sigma_2}(G_{\sigma_1}(s)))\cdots))$$

通过令 $d(n)=r(n)=n$，并应用与 G'起相同作用的函数，可以由构造 3.6.10 重新得到构造 3.6.5。扩充定理 3.6.6 的证明，我们得到如下定理。

定理 3.6.11：设 G,G',f_s 和构造 3.6.10 中一样，并假设 G 是一个伪随机发生器。进一步假设 G'是多项式时间可计算的，总体$\{G'(U_n)\}_{n\in\mathbb{N}}$是定义 3.2.8 中所定义的伪随机总体[①]。则$\{f_s\}_{s\in\{0,1\}^*}$ 是可有效计算的一般的伪随机函数总体。

证明：当 G'是一个恒等变形(且 $r(n)=n$)时，证明与定理 3.6.6 的证明基本相同。为了处理一般情况，我们应用混合证明。特别地，我们应用一个惟一的中间混合(即，函数总体$\{f_s\}$的一个惟一的混合和一个真随机函数)；对每一个 n，我们考虑由 $g(x)=G(h'(x))$定义的(随机)函数 $g:\{0,1\}^{d(n)}\to\{0,1\}^{r(n)}$，其中 h'是从所有由 $d(n)$位长的串映射到 n 位长的串的函数中均匀选择的。只要说明这个混合总体与均匀函数总体以及构造 3.6.10 中的函数总体都是不可分辨的，我们的定理就得证了。

后面，我们用 H_n(或 H_n')表示服从所有由 $d(n)$位长的串映射到 $r(n)$(或 n)位长的串的函数集合上的均匀分布的随机变量。回想前面，混合分布 $G'\circ H_n'$是由固定函数 G'和随机函数分布 H_n'经过函数合成得来的。照常，F_n 表示服从多重集$\{f_s\}_{s\in\{0,1\}^n}$上的均匀分布的随机变量。

结论 3.6.11.1：对于每一个概率多项式时间预言机 M、每一个多项式 $p(\cdot)$和所有充分大的 n，都有

$$|\Pr[M^{G'\circ H_n'}(1^n)=1]-\Pr[M^{H_n}(1^n)=1]|<\frac{1}{p(n)}$$

略证：直观地，预言机访问 $G'\circ H_n'$，相当于赋给它来自分布 $G'(U_n)$的复合独立样本，而预言机访问 H_n 就相当于赋给它来自分布 $U_{r(n)}$的复合独立样本。利用$\{G'(U_n)\}_{n\in\mathbb{N}}$的伪随机性，结论得证。

在实际证明中，我们将预言机 M 转化为得到一个样本序列并模拟 M 的一次运行的一般机 M'，并使用它的输入序列模拟 M 的一些相关预言。特别地，当输入 $\alpha_1,\cdots,\alpha_T$ 时，M'调用 M，并用 α_i 回答它的第 i 个质疑(不失一般性，我们假设 M 不重复处理相同的质疑)。

1. 实际上，当输入来自分布 $G'(U_n)$的样本序列时，M'模拟 $M^{G'\circ H_n'}(1^n)$的一次运行。(关键是观察 $G'\circ H_n'$对不同的质疑序列 $q_1,\ldots,q_t$ 的回答 $G'(s_{q_1}),\ldots,G'(s_{q_t})$，其中每一个 s_{q_i} 相

---

① 当(对所有的 n)都有 $r(n)>n$ 时，我们要求 G'是一个伪随机发生器。但在其他情况就不能这样要求，因为 G'不是展开的。虽然如此，这里我们经常要求伪随机发生器的其他一些性质(即，可有效计算性和输出的伪随机性)。

互独立且服从$\{0,1\}^n$上的均匀分布。)

2. 另一方面，当输入来自分布$U_{r(n)}$的样本序列时，M'模拟$M^{H_n}(1^n)$的一次运行。

(关键是观察H_n预言对于序列$q_1,\ldots,q_t$的不同质疑的回答是服从$\{0,1\}^{r(n)}$的均匀分布。)

因此，如果M不符合上述结论的陈述，那么M'就不符合$\{G'(U_n)\}_{n\in\mathbb{N}}$的伪随机性，这与定理的前提相矛盾。

结论 3.6.11.2：对于每一个概率多项式时间预言机M、每一个多项式$p(\cdot)$和所有充分大的n，都有

$$|\Pr[M^{G',H_n'}(1^n)=1]-\Pr[M^{F_n}(1^n)=1]|<\frac{1}{p(n)}$$

略证：任意函数f_s(构造 3.6.10 中所定义)都可以写为$f_s(x)=G'(f_s'(x))$，其中f_s'如下定义：

$$f_s'(\sigma_1\sigma_2\cdots\sigma_{d(n)})\stackrel{\text{def}}{=}G_{\sigma_{d(n)}}(\cdots(G_{\sigma_2}(G_{\sigma_1}(s)))\cdots)\tag{3.16}$$

我们已经说明$\{f_s'\}$是一个一般的伪随机函数总体(即，f_s'对应于G'是恒等变形的情况)，从而将G'合并到分辨器中，结论得证。

在实际证明中，我们将预言机M变形为预言机M'，它使用自己的预言来模拟一些M的相关预言，特别地，当M处理质疑q时，M'将q转化为自己的预言，把G'应用于它接收到的回答，并将结果转送给M。

1. 实际上，当预言访问h'时，M'模拟$M^{G',h'}(1^n)$的一次运行(原因是在这种情况下，M'以$G'(h'(q))=(G'\circ h')(q))$来回答质疑$q$($M$的质疑)，因此当预言访问$H'_n$时，预言机$M'$模拟$M^{G',H'_n}(1^n)$的一次运行。

2. 另一方面，当预言访问f_s'时，M'模拟$M^{f_s}(1^n)$的一次运行(因为在这种情况下，M'以$G'(f_s'(q))=f_s(q)$来回答M的质疑q)。因此，对均匀选择的$s\in\{0,1\}^n$，当预言访问f'_s时，M'模拟$M^{F_n}(1^n)$的一次运行。

因此，如果M不符结论题的条件，M'就违反$\{f'_s\}$的伪随机性，而这与我们已经建立的理论相矛盾。

结合结论 3.6.11.1 和 3.6.11.2，定理得证。

注释 定理 3.6.11 的证明中，一个主要的部分实际上是证明了如下命题。

设$\{f_s':\{0,1\}^{d(|s|)}\to\{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$是一个一般的伪随机函数总体，$G'$和定理前提中的一样，则由$f_s(x)\stackrel{\text{def}}{=}G'(f_s'(x))$定义的一般的函数总体$\{f_s:\{0,1\}^{d(|s|)}\to\{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$是伪随机的。

结论 3.6.11.2 的证明实际上建立了这个命题，并将它应用到了方程式 3.16 定义的总体$\{f_s'\}_{s\in\{0,1\}^*}$上。

3.6.4.2 定义在全体串上的函数

到目前为止，我们只考虑了有限函数(即，定义域和值域都有限)的函数总体。按理，应用这样的函数就要事先知道这些函数作用域内的输入长度的上界的知识，也就是给定一个函数的应用范围(较短输入可以加密为较长的预先确定长度的输入)。然而，我们更希望对上界的先验知识不作要求(例如，这样的要求可能会排除某些方面的应用)。因此有一个更灵活的伪

随机函数总体的定义是有益的，即允许函数作用于预先不知的变长输入。下面定义并构造这样的总体。

定义 3.6.12(有无界输入的伪随机总体)：设 $r:\mathbb{N}\to\mathbb{N}$。我们称

$$\{f_s:\{0,1\}^*\to\{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$$

是一个可有效计算的有无界输入的伪随机函数总体(有无界输入的伪随机函数)，如果满足下面两个条件：

1. 有效估计：存在一个多项式时间算法，当输入 s 和 $x\in\{0,1\}^*$ 时，返回 $f_s(x)$。

2. 伪随机性：对于每一个概率多项式时间预言机 M、每一个多项式 $p(\cdot)$ 和所有充分大的 n，都有

$$|\Pr[M^{F_n}(1^n)=1]-\Pr[M^{H_n}(1^n)=1]|<\frac{1}{p(n)}$$

其中 F_n 是服从多重集 $\{f_s\}_{s\in\{0,1\}^n}$ 上的均匀分布的随机变量，H_n 服从所有由任意长度的串映射到 $r(n)$ 位长的串的函数上的均匀分布[①]。

关于定义 3.6.12 的几个注释是非常必要的。首先，不知道 f_s 的输入长度理论上在第 1 条中不会引起任何问题，因为估计算法的运行时间可能依赖(多项式关系)于 f_s 的输入长度。至于第 2 条，因为 M 有一个运行时间的理论上界(多项式)，这就从上面限制了对预言机质疑的长度。后一事实解决了前面定义中引起的技术问题(见脚注①)。在典型应用中，我们用 $r(n)=n$(或 $r(n)$ 与 n 多项式相关)。另一个有意义的特殊情况是 $r\equiv 1$ 时的情况，也就是伪随机布尔函数的情况。

与构造 3.6.5 和 3.6.10 类似，对任意 $r:\mathbb{N}\to\mathbb{N}$，满足 $r(n)$ 在 poly(n) 时间内可以由 n 计算得到，利用任意的伪随机发生器可以构造有输入上界的伪随机函数。特别地：

构造 3.6.13：设 G 是将长为 n 的输入扩充为长为 $2n+r(n)$ 的串的确定性算法。记 $G(s)$ 的 $|s|$ 位长的前缀为 $G_0(s)$，$G(s)$ 的第二个 $|s|$ 位的串为 $G_1(s)$，$G(s)$ 的 $r(|s|)$ 位长的后缀记为 $G_2(s)$(即 $G(s)=G_0(s)G_1(s)G_2(s)$)，则对每一个 $s\in\{0,1\}^n$，我们定义一个函数 $f_s:\{0,1\}^*\to\{0,1\}^{r(n)}$，使得对于每一个非负整数 d 和 $\sigma_1,\ldots,\sigma_d\in\{0,1\}$，都有

$$f_s(\sigma_1\sigma_2\cdots\sigma_d)\stackrel{\text{def}}{=}G_2(G_{\sigma_d}(\cdots(G_{\sigma_2}(G_{\sigma_1}(s)))\cdots))$$

由图解，通过沿着有节点标号的无限三重树走的步长来定义函数 f_s。内部节点的标号是 $|s|$ 位长的，叶子节点的标号是 $r(|s|)$ 位长的。根节点后面用来指树的 0 层节点，标号为串 s。如果一个内部节点标号为 s'，那么它最左端的孩子标号为 $G_0(s')$，中间的孩子标号为 $G_1(s')$，最右端的孩子标号为 $G_2(s')$。每一个内部节点的前两个孩子是内部节点，而内部节点的最右端的孩子是一个叶子。当根节点的标号为 s 时，$f_s(\sigma_1\sigma_2\cdots\sigma_d)$ 的值是从根节点"沿着路径 $\sigma_1,\ldots,\sigma_d,2$"可到达的叶子上的串。再者，扩充定理 3.6.6 的证明，我们得到如下结论。

定理 3.6.14：设 G 和所有的 f_s 如构造 3.6.13 中所述，并假设 G 是一个伪随机发生器，则 $\{f_s\}_{s\in\{0,1\}^*}$ 是一个可有效计算的输入无界的伪随机函数总体。

略证：我们遵循定理 3.6.6 的证明方法。也就是说，我们使用混合技术，其中将分配第 k

---

① 因为 M 的运行时间在理论上以一些多项式为界，所以对一些多项式 d 和所有的 n，当输入 1^n 时，预言机 M 的质疑长度至多为 $d(n)$。因此，可以定义 H_n 为由长度以 $d(n)$ 为上界的串映射到 $r(n)$ 位的串的所有函数上的均匀分布。这就解决了无限集(即，所有由任意位长的串映射到 $r(n)$ 位长的串上的函数集合)上均匀分布想要解决的技术问题。

个混合给这样的函数，此函数为树的最高的 $k+1$ 层节点均匀选择标号，并为构造 3.6.13 中的较低层计算标号。特别地，定义第 k 个混合为函数 $f_{s_1,\ldots,s_{3^k}}:\{0,1\}^* \to \{0,1\}^{r(n)}$，此函数下面定义，其中 $s_1,\ldots,s_{3^k}\in\{0,1\}^{2n+r(n)}$ 相互独立且服从均匀分布。

$$f_{s_1,\ldots,s_{3^k}}(\sigma_1\sigma_2\cdots\sigma_d)$$

$$\stackrel{\text{def}}{=}\begin{cases}P_2(s_{\mathrm{idx}(2^{k-d}\cdot\sigma_d\cdots\sigma_1)}) & d\leqslant k\\ G_2(G_{\sigma_d}(\cdots(G_{\sigma_{k+2}}(G_{\sigma_{k+1}}(s_{\mathrm{idx}(\sigma_k\cdots\sigma_1)})))\cdots)) & \text{其他}\end{cases}$$

这里 $\mathrm{idx}(\alpha)$ 是长度为 $|\alpha|$ 的三元串中按标准字典顺序排列的 α 的索引，$P_2(\beta)$ 是 β 的 $r(n)$ 位长的后缀。

注意，（与定理 3.6.6 的证明不同）对每一个 n 都有无限多的混合，因为这里的 k 可以是任意非负整数（而不是定理 3.6.6 中的 $k\in\{0,1,\ldots,n\}$）。并且，因为我们考虑的是（任意）概率多项式时间分辨器 M，就存在多项式 d，使得当输入 1^n 时，预言机 M 的质疑长度至多为 $d(n)-1$。因此，预言机 M 访问第 $d(n)$ 个混合，就等价于 M 访问均匀随机变量 H_n（这里 H_n 和定义 3.6.12 中一样），因为对 $i\leq d(n)$，每一个 i 层的叶子都分配了一个均匀选择的标号。另一方面，第 0 个混合与随即变量 F_n 相对应（这里 F_n 和定义 3.6.12 中一样），因为根节点也分配了一个均匀选择的标号。这样，如果 M 能够分辨 $\{H_n\}$ 和 $\{F_n\}$，它就能分辨一对（随机的）相邻混合（即第 $k-1$ 和第 k 个混合，其中 k 是从 $\{1,\ldots,d(n)\}$ 中均匀选择的）。像定理 3.6.6 的证明中一样，后一个声明可以用来推翻 G 的伪随机性。特别地，我们可以分辨来自分布 $U_{2n+r(n)}$ 和来自分布 $G(U_n)$ 的复合独立样本：给定 $(2n+r(n))$ 位长的串序列，用它们来为树的最高 $k+1$ 层节点标号（将每一个串分为三部分，对于 $i\leq k$，用这三部分为 $i-1$ 层的节点的孩子标号）。当串来自 $U_{2n+r(n)}$ 时，我们模拟第 k 个混合，而当串来自 $G(U_n)$ 时，我们模拟第 $k-1$ 个混合。从而定理得证。

注释　无界输入（并一般化）的伪随机函数可由（一般的）伪随机函数直接构造；详见 3.8.2 小节。

3.7*　伪随机置换

本节我们给出了伪随机置换的定义和构造。显然，（在极大域上的）伪随机置换在任何有效的应用中都可以代替伪随机函数，且伪随机置换具有包含惟一的原像这一优点。虽然这一额外的优点某些时候是有用的，但是其作用可能比人们的期望值要小（例如，本书余下的部分都不使用它，甚至有关加密机制的章节中也不使用，其原因在后面将给出）。

我们将说明如何按照一种等价于 DES 的高层构造的方式，把伪随机函数作为构造块来构造伪随机置换。因此，本节给出的证明过程可以被看作支持 DES 方法的过程，此方法把“看似随机”的函数转化为“看似随机”的置换。[①]

---

① 在 DES 中此方法作用于那些非“看似随机的”函数的情况在这里不予考虑。

3.7.1 一些定义

首先我们定义伪随机置换。简单地说，定义置换的伪随机总体的方法和定义函数的伪随机总体的方法类似。即：

定义 3.7.1(置换总体)：**置换总体**就是一个随机变量的序列 $P=\{P_n\}_{n\in\mathbb{N}}$，其中随机变量 P_n 的取值为把长为 n 的字符串映射到长为 n 的字符串的置换。均匀置换总体记为 $K=\{K_n\}_{n\in\mathbb{N}}$，其中 K_n 服从把长为 n 的字符串映射为长为 n 的字符串的置换集合上的均匀分布。

每个置换总体都是函数总体，因此可被有效计算的置换总体的定义是显而易见的(即可以根据可被有效计算的函数总体的定义得到)。伪随机置换可定义为与均匀置换总体计算不可分辨的总体。

定义 3.7.2(伪随机置换总体)：如果对于每个概率多项式时间图灵机 M、每个多项式和所有足够大的 n，某一置换总体 $P=\{P_n\}_{n\in\mathbb{N}}$有

$$|\mathrm{Pr}[M^{P_n}(1^n)=1]-\mathrm{Pr}[M^{K_n}(1^n)=1]|<\frac{1}{p(n)}$$

其中 $K=\{K_n\}_{n\in\mathbb{N}}$是均匀置换总体，则此置换总体 P 就是伪随机的。

给定输入 P_n，观测机并不能在 poly(n)时间内检测出 P 是一个伪随机置换总体，而非仅仅是伪随机函数总体。这一事实源于如下结论：均匀置换总体和均匀函数总体在多项式时间内是不可分辨的。即：

命题 3.7.3：均匀置换总体(即 $K=\{K_n\}_{n\in\mathbb{N}}$)构成伪随机函数总体。

略证：$\{H_n\}_{n\in\mathbb{N}}$表示均匀函数总体，当给定预言 H_n 时，机器能检测出预言函数中冲突的概率上界为 $t^2\cdot 2^{-n}$，其中 t 表示此机器发出的质疑数。如果没有发现这样的冲突，那么 H_n 的响应和 K_n 的响应是不可分辨的。最后，根据一个多项式机器最多能发起多项式个质疑，命题得证。

因此，用伪随机置换代替伪随机函数的原因并不在于一个计算上有限的观察机是否能检测出二者的不同的问题。典型地，其原因在于人们想保证原像的惟一性。对这个要求的一种很自然的加强方法是给出置换的描述，就可以有效地找到惟一的原像。

定义 3.7.4(可有效计算和求逆的置换总体)：一个置换总体 $P=\{P_n\}_{n\in\mathbb{N}}$如果满足以下 3 个条件：

1. 有效索引：存在一个概率多项式时间算法 I 和一个从字符串到置换 ϕ 的映射，使得 $\phi(I(1^n)$和 P_n 的分布等价。

2. 有效求值：存在一个概率多项式时间算法 V，使得 $V(i,x)=f_i(x)$，其中(如定义 3.6.3 中所述)$f_i\overset{def}{=}\phi(i)$。

3. 有效求逆：存在一个概率多项式时间算法 N，使得 $N(i,x)=f_i^{-1}(x)$(即 $f_i(N(i,x))=x$)。

则此置换总体是可有效计算和求逆的。

第 1 和第 2 项可以通过可有效计算的置换总体的定义来保证。第 3 项则是附加条件。在某些情况下，扩大伪随机总体是有意义的，其扩大方法是要求此总体在给观察机输入两个预言——一个是置换而另一个是它的逆置换时，观察机不能分辨此总体和均匀总体。因此，我们认

为扩充的预言机可以向两个预言提出质疑；这种双预言模型可以由标准的单预言模型来模拟，通过将两个预言 f_1 和 f_2 合成为一个预言 $f(f(i,q))=f_i(q)$。

定义 3.7.5(严格伪随机置换)：如果对于每个概率多项式时间预言机 M、每个多项式 $p(\cdot)$ 和所有足够大的 n 而言，某一置换总体 $P=\{P_n\}_{n\in\mathbb{N}}$ 有

$$\left|\Pr[M^{P_n,P_n^{-1}}(1^n)=1]-\Pr[M^{K_n,K_n^{-1}}(1^n)=1]\right|<\frac{1}{p(n)}$$

其中 $M^{f,g}$ 表示输入预言 f 和 g 时机器 M 的行为，则称此置换总体是**严格伪随机的**。

3.7.2 构造

伪随机置换的构造使用伪随机函数作为构造块，与 DES 的高层构造方式相同(见图 3.6)即，

构造 3.7.6：令 $f:\{0,1\}^n\to\{0,1\}^n$。对每个 $x,y\in\{0,1\}^n$，我们定义

$$\mathrm{DES}_f(x,y)\stackrel{\mathrm{def}}{=}(y,x\oplus f(y))$$

其中 $x\oplus y$ 表示二进制字符串 x 和 y 的逐位异或 XOR、类似地，对每个 $f_1,\ldots,f_t:\{0,1\}^n\to\{0,1\}^n$，我们定义

$$\mathrm{DES}_{f_t,\ldots,f_1}(x,y)\stackrel{\mathrm{def}}{=}\mathrm{DES}_{f_t,\ldots,f_2}(\mathrm{DES}_{f_1}(x,y))$$

对每个函数总体 $F=\{F_n\}_{n\in\mathbb{N}}$ 和每个函数 $t:\mathbb{N}\to\mathbb{N}$，我们通过令 $\mathrm{DES}_{F_n}^{t(n)}\stackrel{\mathrm{def}}{=}\mathrm{DES}_{F_n^{(t)},\ldots,F_n^{(1)}}$

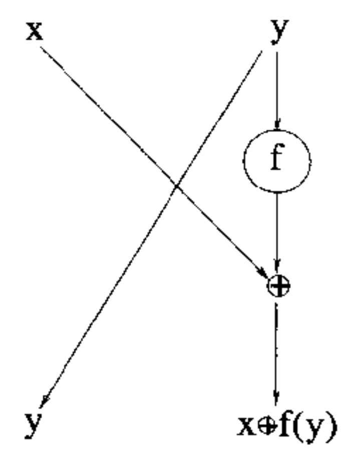


图 3.6 DES 的高层构造

来定义函数总体 $\{\mathrm{DES}_{F_n}^{t(n)}\}_{n\in\mathbb{N}}$，其中 $t=t(n)$，$F_n^{(i)}$ 是随机变量 F_n 的独立副本。

定理 3.7.7：设 F_n，$t(\cdot)$ 和 $\mathrm{DES}_{F_n}^{t(n)}$ 如构造 3.7.6 所述。假设 $\{F_n\}_{n\in\mathbb{N}}$ 是可有效计算的，且当输入 n 时可以在 poly(n)时间内计算 $t(n)$，则对每个多项式时间内可计算的函数 $t(\cdot)$，总体 $\{\mathrm{DES}_{F_n}^{t(n)}\}_{n\in\mathbb{N}}$ 是可有效计算和求逆的置换总体。而且，若 $F=\{F_n\}_{n\in\mathbb{N}}$ 是一个伪随机函数总体，则总体 $\{\mathrm{DES}_{F_n}^{3}\}_{n\in\mathbb{N}}$ 是伪随机的，总体 $\{\mathrm{DES}_{F_n}^{4}\}_{n\in\mathbb{N}}$ 是严格伪随机的。

总体 $\{\mathrm{DES}_{F_n}^{t(n)}\}_{n\in\mathbb{N}}$ 显然是可有效计算的。它是置换总体且具有有效求逆算法这一事实可以遵循如下结论：$\mathrm{DES}_{zero,f,zero}$ 是 DES_f 的逆，其中对所有 $z\in\{0,1\}^n$ 有 $\mathrm{zero}(z)\stackrel{\mathrm{def}}{=}0^{|z|}$。也就是说，对每个 $x,y\in\{0,1\}^n$，$\mathrm{DES}_{zero}(x,y)=(y,x)$ 及

$$\mathrm{DES}_{zero,f,zero}(\mathrm{DES}_f(x,y))=\mathrm{DES}_{zero,f,zero}(y,x\oplus f(y))$$

$$= \mathrm{DES}_{zero,f}(x \oplus f(y), y)$$
$$= \mathrm{DES}_{zero}(y, (x \oplus f(y)) \oplus f(y))$$
$$= (x, y)$$

要证明$\{\mathrm{DES}^3_{F_n}\}_{n\in\mathbb{N}}$的伪随机性(或$\{\mathrm{DES}^4_{F_n}\}_{n\in\mathbb{N}}$的严格伪随机性)，就必须证明$\{\mathrm{DES}^3_{H_n}\}_{n\in\mathbb{N}}$的伪随机性(或$\{\mathrm{DES}^4_{H_n}\}_{n\in\mathbb{N}}$的严格伪随机件)。其原因是，若$\{\mathrm{DES}^3_{H_n}\}_{n\in\mathbb{N}}$是伪随机的，而$\{\mathrm{DES}^3_{F_n}\}_{n\in\mathbb{N}}$不是伪随机的，则会得到一个和函数总体 F 的伪随机性相矛盾的结论(即，可以把 F 和均匀函数总体 H 区分开来；参见习题 35)。因此，定理 3.7.7 是根据命题 3.7.8 得出的。

命题 3.7.8：$\{\mathrm{DES}^3_{H_n}\}_{n\in\mathbb{N}}$是伪随机的，而$\{\mathrm{DES}^4_{H_n}\}_{n\in\mathbb{N}}$是严格伪随机的。

略证：首先证明$\{\mathrm{DES}^3_{H_n}\}_{n\in\mathbb{N}}$是伪随机的。令 $P_{2n} \stackrel{\text{def}}{=} \{\mathrm{DES}^3_{H_n}\}_{n\in\mathbb{N}}$，$K_{2n}$是在作用于$\{0,1\}^{2n}$上的所有可能置换上均匀分布的随机变量。可以证明，对每个输入 1^n 时最多能提出 m 个质疑的的预言机 M，下式成立

$$|\Pr[M^{P_{2n}}(1^n) = 1] - \Pr[M^{K_{2n}}(1^n) = 1]| \leq \frac{2m^2}{2^n} \tag{3.17}$$

令 $q_i = (L_i^0, R_i^0)$(其中 $|L_i^0| = |R_i^0| = n$)为一随机变量，表示当给定预言 P_{2n}时 M 的第 i 个质疑。已知 $P_{2n} = \mathrm{DES}_{H_n^{(3)}, H_n^{(2)}, H_n^{(1)}}$，其中这些 $H_n^{(j)}$ 表示 3 个相互独立的随机变量，每个都在作用于$\{0,1\}^n$ 上的函数上均匀分布。令 $R_i^{k+1} \stackrel{\text{def}}{=} L_i^k \oplus H_n^{(k+1)}(R_i^k)$，并对 $k=0,1,2$ 定义 $L_i^{k+1} \stackrel{\text{def}}{=} R_i^k$。于是可得，

$$(L_i^{k+1}, R_i^{k+1}) = (R_i^k, L_i^k \oplus H_n^{(k+1)}(R_i^k))$$

不失一般性，我们假设 M 不会提出两个相同的质疑。定义一个随机变量 ζ_m 来表示如下事件：存在 $i<j\leq m$ 和 $k\in\{1,2\}$，使得 $R_i^k = R_j^k$(即，当输入 1^n 和预言 P_{2n}时，M 的前 m 个质疑中的任意两个质疑都满足 $R_i^k = R_j^k$ 这一关系)。下面是我们要使用到的两个结论。

论题 3.7.8.1：在 $\neg\zeta_m$ 的条件下，对于每个 $m\geq 1$，R_i^3 是在$\{0,1\}^n$ 上均匀独立分布的，而 L_i^3 是在尚未分配给前面的 L_j^3 的长为 n 的字符串集上均匀分布的。即，对于每个 $\alpha_1, \ldots, \alpha_m \in \{0,1\}^n$，有

$$\Pr\left[\wedge_{i=1}^m (R_i^3 = \alpha_i) \mid \neg\zeta_m\right] = \left(\frac{1}{2^n}\right)^m \tag{3.18}$$

而对于每个不同的 $\beta_1, \ldots, \beta_m \in \{0,1\}^n$，有

$$\Pr\left[\wedge_{i=1}^m (L_i^3 = \beta_i) \mid \neg\zeta_m\right] = \prod_{i=1}^m \frac{1}{2^n - i + 1} \tag{3.19}$$

证明思路：等式 3.18 是由如下结论得到的：通过将随机函数 $H_n^{(3)}$ 作用于不同的自变量(即所有的 R_i^2)，可以确定 R_i^3，其中 $\neg\zeta_m$ 就隐含了所有 R_i^2 的互异性。类似地，通过将随机函数 $H_n^{(2)}$ 作用于不同的自变量(即不同的 R_i^1)，可以确定 $L_i^3 = R_i^2$，$\neg\zeta_m$ 也是结果(即所有的 R_i^2)互异的条件。于是等式 3.19 成立。

结论 3.7.8.2：对于每个 $m\geq 1$，有

$$\Pr[\zeta_{m+1} \mid \neg\zeta_m] \leq \frac{2m}{2^n}$$

证明思路：选定任意的 $i\leq m$，考虑 $R_{m+1}^1 = R_i^1$ 的概率。分两种情况考虑：

1. 若 $R_i^0 = R_{m+1}^0$，则一定有(因为$(L_i^0, R_i^0) \neq (L_{m+1}^0, R_{m+1}^0)$)

$$R_i^1 = L_i^0 \oplus H_n^{(1)}(R_i^0) = L_i^0 \oplus H_n^{(1)}(R_{m+1}^0) \neq L_{m+1}^0 \oplus H_n^{(1)}(R_{m+1}^0) = R_{m+1}^1$$

2. 另一方面，若 $R_i^0 \neq R_{m-1}^0$，则

$$\Pr[R_i^1 = R_{m+1}^1] = \Pr[H_n^{(1)}(R_i^0) \oplus H_n^{(1)}(R_{m+1}^0) = L_i^0 \oplus L_{m+1}^0] = 2^{-n}$$

其中最后一个等式成立的原因是：随机函数 $H_n^{(1)}$ 作用于不同的自变量（即 R_i^0 和 R_{m+1}^0）。

于是，对两种情况都有 $\Pr[R_i^1 = R_{m+1}^1] \leq 2^{-n}$。和前面第 2 种情况类似，当 $R_i^1 \neq R_{m-1}^1$ 时，我们可以得到

$$\Pr[R_i^2 = R_{m-1}^2] = \Pr[H_n^{(2)}(R_i^1) \oplus H_n^{(2)}(R_{m-1}^1) = R_i^0 \oplus R_{m-1}^0] = 2^{-n}$$

从而对每个 $i \leq m$，有

$$\Pr[R_i^1 = R_{m+1}^1 \vee R_i^2 = R_{m+1}^2] \leq \Pr[R_i^1 = R_{m+1}^1] + \Pr[R_i^2 = R_{m+1}^2 \mid R_i^1 \neq R_{m+1}^1]$$
$$\leq 2 \cdot 2^{-n}$$

结论得证。

根据 $\Pr[\zeta_m] \leq \Pr[\zeta_{m-1}] + \Pr[\zeta_m \mid \neg\zeta_{m-1}]$ 和结论 3.7.8.2，对 m 用归纳法可以证明 $\Pr[\zeta_m] < \frac{m^2}{2^n}$。根据结论 3.7.8.1，在 $\neg\zeta_m$ 的条件下，P_{2n} 的响应的左半部分是在所有不同于前面的响应的左半部分的长为 n 的字符串上均匀分布的，而其右半部分是在所有长为 n 的字符串上均匀分布的。另一方面，K_{2n} 的响应是在所有不同于前面的响应的长为 $2n$ 的字符串上均匀分布的。因此，这两种情况下所有响应的分布之间的统计差异不能超过 $\frac{m^2}{2^n} + \binom{m}{2} \cdot 2^{-n} < \frac{m^2}{2^n}$，等式 3.17 成立。

$\{\mathrm{DES}_{H_n}^4\}_{n\in\mathbb{N}}$ 是严格伪随机的证明过程更加复杂，但是其思路本质上是相同的。[①] 特别地，与 ζ_m 相对应的事件是四类事件的分离。第一类事件对 $k \in \{2,3\}$ 有 $R_i^k = R_j^k$，其中 $q_i = (L_i^0, R_i^0)$ 和 $q_j = (L_j^0, R_j^0)$ 是前向的质疑。类似地，第二类事件对 $k \in \{2,1\}$ 有 $R_i^k = R_j^k$，其中 $q_i = (L_i^4, R_i^4)$ 和 $q_j = (L_j^4, R_j^4)$ 是后向的质疑。第三类事件对 $k \in \{2,3\}$ 有 $R_i^k = R_j^k$，其中 $q_i = (L_i^0, R_i^0)$ 是前向的质疑，$q_j = (L_j^4, R_j^4)$ 是后向的质疑，且 $j < i$。类似地，第四类事件对 $k \in \{2,1\}$ 有 $R_i^k = R_j^k$，其中 $q_i = (L_i^4, R_i^4)$ 是后向的质疑，$q_j = (L_j^0, R_j^0)$ 是前向的质疑，且 $j < i$。和前面类似，可以限制事件 ζ_m 的概率，以及假定 ζ_m 为假时 K_{2n} 的响应和 $\{\mathrm{DES}_{H_n}^4\}_{n\in\mathbb{N}}$ 的响应之间的统计差异。■

3.8 其　他

3.8.1 历史记录

Goldwasser 和 Micali（在加密机制中）首先提出了计算上不可分辨的总体的概念[123]。在一般机制中，这一概念首先出现在 Yao 的著作中，它也是伪随机性的定义的雏形[210]。Yao 也证明了伪随机总体可能和均匀总体大大不同，然而我们对命题 3.2.3 的证明却来自[107]。

---

① 这里，我们假设智能机避免那些它知道答案的质疑。即，不仅它不两次发出相同的质疑，而且如果它发出前向（或后向）质疑 q，并且得到答案 a，那么它将不会发出后向（或前向）质疑 a。

Blum 和 Micali[36]提出了伪随机生成器的概念,他们把这样的生成器定义为产生不可预测序列的算法。Blum 和 Micali 证明了若假设离散对数问题的棘手性,则这样的发生器的确存在。他们进一步给出一个构造伪随机发生器的通用范例,这在后来的研究中都直接或间接地得到应用。不久以后,Goldwasser 等人在[126]中,Blum 等人在[32]中分别提出了其他有关伪随机发生器的建议。随后,Yao 又证明出任何单向置换的存在性隐含了伪随机发生器的存在性[210]。Yao 是把伪随机发生器定义为产生计算上和均匀序列不可分辨的序列的第一人。他还证明出这个伪随机发生器的定义等价于 Blum 和 Micali 的定义[36]。

Levin[150]和 Goldreich,Krawczyk,Luby[108]把 Yao 的结论概括为单项置换隐含了伪随机发生器,并出版,最后终于得到 Håstad,Impagliazzo,Levin 和 Luby 的结论[129]:伪随机发生器存在当且仅当单向函数存在。3.5 节给出的构造正是遵循这些理论的[108,129]。这些构造利用了大量的通用$_2$哈希函数,这一函数是 Carter 和 Wegman 提出的[49],Sipser 首次将它用于复杂性理论[201]。

基于专门的复杂性假设的简单伪随机发生器在[36,32,5,208,141]中给出。特别地,[5]提出了基于因数分解的伪随机发生器,而[141]提出了基于不同离散对数问题(参见 2.4.3.4 节)的伪随机发生器。这两种方法的主要步骤是构造对应的单向置换集的核心断言。

伪随机函数由 Goldreich,Goldwasser 和 Micali 提出并研究[102]。特别地,基于伪随机发生器的伪随机函数的构造就来自于[102]。伪随机函数的首次应用在[103,89,90]中给出,继而这一系列的应用就越来越多了。

Luby 和 Rackoff 定义并构造了伪随机置换[156],我们提出的伪随机置换就参照了他们的著作。

混合方法来源于 Goldwasser 和 Micali 的著作[123]。术语是由 Leonid Levin 提出的。

3.8.2 关于进一步阅读的建议

Goldreich 给出广义上的伪随机性[97]。他分析了各种不同的伪随机发生器的概念,把本章讨论的这一概念视为通用范例的一个典型实例。通用范例就是把那些利用某种资源有限的分辨器无法和均匀分布区分开的分布视为伪随机的。发生器本身的复杂性和它的扩展函数一样是变化的(而非像这里所说的那样受多项式时间和多项式约束)。从通用范例开始,[97]的第 3 章分析了伪随机发生器的典型情况(就是我们这里所考虑的)和能抗空间约束的识别器的发生器,诸如 ***BPP*** 的复杂度类的去随机性,以及各种特殊用途的发生器。(对 Kolmogorov 复杂性感兴趣的读者请参考[152]。)

命题 3.2.3 给出了一对计算上不可分辨的总体,尽管它们统计上差别很大。虽然不用做任何复杂性的假设就能证明它,但是这两个总体的其中之一是在多项式时间内不可构造的。这种情况是无法避免的,因为具有此种属性(即计算不可分辨却有很大的统计差异)的一对多项式时间内可构造的总体的存在性隐含了单向函数的存在性[93]。其他有关计算上不可分辨这一概念的抽象结论在[111,119]中给出。

结合定理 2.5.6 和构造 3.4.7,可以得到一个伪随机发生器的一般(黑盒)构造,这是基于任意的在每次应用时都能输出长为对数的比特串的单向置换的构造。其他地方[88]还指出,只要一般(黑盒)构造有效,则它就能达到我们所能期望的最佳性能(即每次运用此单向函数时

输出对数多的比特数)。

3.5 节没有给出Håstad等人[129]提出的构造,也没有证明它的有效性。不幸的是,这个基本定理　断言若单向函数存在则伪随机发生器也存在——的证明太复杂而不适合本书。感兴趣的读者可以参考原始文献[129]。

伪随机函数的另--构造在[172]中给出。基于(普通)伪随机函数构造无限输入(并一般化的)伪随机函数的方法在[14]中讨论。

[173]介绍了另一种(基于伪随机函数的)伪随机置换的构造。它吸取了此证明过程的实际构造的精华,并给出了相应的结论。

伪随机发生器和伪随机函数在密码学上有诸多应用;其中一些应用将在本书第二卷中给出(例如,数字签名和加密)。

使用不完全随机的信源　虽然伪随机发生器和函数使我们能够扩展随机性(或伪随机性),但是我们并不能"产生确定的随机性(或伪随机性)"。实际上,我们不能期望获得一个有效的能产生伪随机对象的确定性程序(因为同样的程序可能被分辨器使用)。要使用伪随机发生器(函数),首先必须有一个随机种子,那么如何获得这一种子呢?通过采样某一物理现象可以得到随机种子(或看似随机的)。当然,这些采样可能不是在(特定长度的)字符串的集合上均匀分布的,但是如果它们包含足够的信息量,那么就能从中(有效地)提取出几乎完全的随机性。这种随机性的提取方法将在本书第三卷中讨论。

随机预言方法论　与 3.6.3 小节中讨论的方法相比,随机预言模型是指一种敌方可直接访问随机预言(它是通过一个函数来"实现"的,函数的描述敌方也知道)的装置。随机预言方法论[80,21]首先设计了一个理想系统,在此系统中每一方(包括敌方)都可以直接访问一下真正随机的预言函数,然后用"好的哈希函数"代替随机预言,并为每一方(包括敌方)提供此函数的简单表达式。相反,3.6.3 小节的方法论是指这一情形:敌方不能直接访问预言随机函数,也不能获得后面的实现中用到的伪随机函数的表达式。值得注意的是,与 3.6.3 小节提出的方法论相反,随机预言方法论是启发式的。特别地,在随机预言模型中,安全的加密机制和签名机制是存在的,但没有任何安全的通过函数总体来实现的方法。

3.8.3　未决问题

虽然Håstad等人[129]说明了如何利用任意给定的单向函数构造伪随机发生器,但是他们的构造是不切实际的,其原因是输入长为 n 的种子时,发生器的"性能"与输入长度小于$\sqrt[4]{n}$时对给定函数求逆的困难度相关利用。我们认为,提出一种有效转换,使之能把任意单向函数转化为伪随机发生器,是这一领域最重要的未决问题之一要解决这一问题,需要发现新的重要范例。

更具实际意义的未决问题则是:构造更有效的伪随机发生器,基于特定计算问题(例如整数的因数分解)的复杂性。详情请参见 3.4.3 和 2.7.3 小节。

3.8.4　习题

习题 1:计算上不可分辨性,细微变化:试证明,对定义 3.2.2 分别做如下细微的改动,它

们都是等价的。其中总体 X 和 Y 都满足如下定义：$X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathbb{N}}$ 和 $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathbb{N}}$。

1. 如果对每个概率多项式时间算法 D、每个正多项式 $p(\cdot)$ 和所有足够大的 n，有

$$|\Pr[D(X_n,1^n)=1]-\Pr[D(Y_n,1^n)=1]| \le \frac{1}{p(n)}$$

那么总体 X 和 Y 是在多项式时间内一类不可分辨的(这里用≤代替严格不等号)。

2. 如果对每个概率多项式时间算法 D、每个正多项式 $p(\cdot)$ 和所有足够大的 n，有

$$\Pr[D(X_n,1^n)=1]-\Pr[D(Y_n,1^n)=1] < \frac{1}{p(n)}$$

那么总体 X 和 Y 是在多项式时间内二类不可分辨的(这里去掉绝对值)。

3. 设 $|X_n|=|Y_n|=n$。如果对每个概率多项式时间算法 D、每个正多项式 $p(\cdot)$ 和所有足够大的 n，有

$$|\Pr[D(X_n)=1]-\Pr[D(Y_n)=1]| < \frac{1}{p(n)}$$

那么总体 X 和 Y 是在多项式时间内三类不可分辨的(这里省略了辅助输入 1^n)。

习题 2：有效算法保持计算上不可分辨性：设 $\{X_n\}_{n\in\mathbb{N}}$ 和 $\{Y_n\}_{n\in\mathbb{N}}$ 是两个多项式时间不可分辨的总体。

1. 试证明，对任何概率多项式时间算法 A，总体 $\{A(X_n)\}_{n\in\mathbb{N}}$ 和 $\{A(Y_n)\}_{n\in\mathbb{N}}$ 是多项式时间不可分辨的。

2. 试说明，若 A 不是多项式时间算法，则总体 $\{A(X_n)\}_{n\in\mathbb{N}}$ 和 $\{A(Y_n)\}_{n\in\mathbb{N}}$ 不一定是多项式时间不可分辨的。

习题 3：任何函数保持统计接近性：设 $\{X_n\}_{n\in\mathbb{N}}$ 和 $\{Y_n\}_{n\in\mathbb{N}}$ 是两个统计上接近的总体，并设 $f: \{0,1\}^* \to \{0,1\}^*$ 为一函数。试证明总体 $\{f(X_n)\}_{n\in\mathbb{N}}$ 和 $\{f(Y_n)\}_{n\in\mathbb{N}}$ 也是统计上接近的。

习题 4：试证明，对于每个 $L\in$ ***BPP*** 和每对多项式时间不可分辨的总体 $\{X_n\}_{n\in\mathbb{N}}$ 和 $\{Y_n\}_{n\in\mathbb{N}}$，函数

$$\Delta_L(n) \stackrel{\text{def}}{=} |\Pr[X_n\in L]-\Pr[Y_n\in L]|$$

对 n 来说是可忽略不计的。

虽然考虑反方向也成立是很吸引人的问题，但是我们还不能证明它是否真的成立；需要注意的是，虽然 $\{X_n\}$ 和 $\{Y_n\}$ 可以用概率算法来区分，但不能被确定性算法区分。此时，我们该定义何种语言呢？例如，假设 A 是一个概率多项式时间算法，并设 $L \stackrel{\text{def}}{=} \{x: \Pr[A(x)=1] \ge \frac{1}{2}\}$，那么 L 不一定要在 ***BPP*** 里。(习题 5 说明在不可计算的装置中，上述结论及其逆命题都是成立的。)

习题 5：统计接近性的一种等价说法：试证明，两个总体 $\{X_n\}_{n\in\mathbb{N}}$ 和 $\{Y_n\}_{n\in\mathbb{N}}$ 是统计上接近的当且仅当对每个集合 $S\subseteq\{0,1\}^*$，

$$\Delta_S(n) \stackrel{\text{def}}{=} |\Pr[X_n\in S]-\Pr[Y_n\in S]|$$

对 n 来说是可忽略不计的。

提示：试证明，如 3.1 式所定义的 X_n 和 Y_n 间的统计差异等于 $\max_S\{\Delta_S(n)\}$。

习题 6：统计接近性隐含着计算上的不可分辨性：试证明，如果两个总体是统计上接近的，那么它们也是多项式时间不可分辨的。

提示：利用习题 5 的结论，并为每个函数 $f: \{0,1\}^* \to \{0,1\}$ 定义一个集合 $S_f \stackrel{\text{def}}{=} \{x:$

$f(x)=1\}$。

习题 7：定理 3.2.6 的信息论类比：试证明，如果两个总体是统计上接近的，那么它们的多项式乘积一定也是统计上接近的。

提示：试证明，两个分布的 m 乘积之间的统计差异不大于这两个分布之间的距离的 m 倍。

习题 8：电路的计算上不可分辨性，概率性（非确定性）和确定性：设 $\{X_n\}_{n\in\mathbb{N}}$ 和 $\{Y_n\}_{n\in\mathbb{N}}$ 是两个总体，并令 $C\stackrel{\text{def}}{=}\{C_n\}_{n\in\mathbb{N}}$ 是一个概率多项式大小的电路族。试证明，存在一个（确定性的）多项式大小的电路族 $D\stackrel{\text{def}}{=}\{D_n\}_{n\in\mathbb{N}}$，使得对每个 n，有

$$\Delta_D(n) \geq \Delta_C(n)$$

其中，

$$\Delta_D(n) \stackrel{\text{def}}{=} |\Pr[D_n(X_n) = 1] - \Pr[D_n(Y_n) = 1]|$$

$$\Delta_C(n) \stackrel{\text{def}}{=} |\Pr[C_n(X_n) = 1] - \Pr[C_n(Y_n) = 1]|$$

习题 9：电路的计算上不可分辨性，单个采样和多个采样：试证明，$X=\{X_n\}_{n\in\mathbb{N}}$ 和 $Y=\{Y_n\}_{n\in\mathbb{N}}$ 是无法用多项式大小的电路（如定义 3.2.7 所述）区分的，当且仅当对于每个多项式 $m(\cdot)$，它们的 $m(\cdot)$-乘积是无法用多项式大小的电路区分的。需要强调的是，X 和 Y 不一定是多项式时间内可构造的。

提示："合理选择"的 $x^1,\ldots,x^k$ 和 $y^{k+2},\ldots,y^m$ 可被"牢固地接到"电路上。

习题 10：计算上不可分辨性，电路和算法：

1.（简单的）假设总体 $X=\{X_n\}_{n\in\mathbb{N}}$ 和 $Y=\{Y_n\}_{n\in\mathbb{N}}$ 是无法用多项式大小的电路区分的。试证明，它们也是（利用概率多项式时间算法）计算上不可分辨的。

提示：利用习题 8。

2.（困难的）试证明，（利用概率多项式时间算法）计算上不可分辨的但用多项式大小的电路可区分的总体是存在的。

提示（第 2 部分）：对任意给定的函数 $f:\{0,1\}^*\to[0,1]$，证明存在一个总体 $X=\{X_n\}_{n\in\mathbb{N}}$，使得每个 X_n 的支集（*support*）的大小最多是 2，且 $\Pr[f(X_n)=1]=\Pr[f(U_n)=1]$，其中 U_n 在 $\{0,1\}^n$ 上均匀分布。把证明一般化，给定 t 个这样的函数 $f_1,\ldots,f_t:\{0,1\}^*\to[0,1]$，其中每个 X_n 支集的大小最多是 $t+1$，且对每个 $i=1,\ldots,t$，有 $\Pr[f_i(X_n)=1]=\Pr[f_i(U_n)=1]$（附加提示：考虑关于每个 $x\in\{0,1\}^n$ 的 t 维向量 $(f_1(x),\ldots,f_t(x))$ 和凸集）。标准对角证明将完成此项工作（如果不知道此证明方法，请参考[111]）。

习题 11：试证明，计算上不可分辨且非统计上接近的一对多项式时间内可构造的总体的存在性隐含了单向函数的存在性。

提示：可以用一种比以前[93]提出的更简单的证明方法。在[93]中证明了这个前提假设隐含了伪随机发生器的存在性。这一证明方法的主要思想还是要用到的：获得每个总体足够多的独立副本，构造两个计算上不可分辨的总体，这两个总体"几乎是没有交集的"（即它们的统计差异至少是 $1-2^{-n}$）。接着，暂且假设这两个总体是无交集的（即统计差异为 1），然后（利用命题 3.3.8 的证明过程中的一个变量）证明本习题的结论。最后，利用类比法处理一般情况，以证明此假设隐含了"分布式单向函数"（参见第 2 章习题 17）的存在性。

习题 12：试证明，伪随机发生器不会把显著概率集分配给任何字符串。即，若 G 是一个伪

随机发生器，则对每个正多项式 p 和所有充分大的 n 和 α，有 $\Pr[G(U_n)=\alpha]<1/p(n)$。

习题 13：伪随机发生器可以产生 1-1 映射吗？也就是说，如果 G 是伪随机发生器，那么映射 $G:\{0,1\}^n\rightarrow\{0,1\}^{l(n)}$ 是 1-1 的吗？

1. 试证明，若伪随机发生器存在，则存在伪随机发生器 G，使得映射 $G:\{0,1\}^n\rightarrow\{0,1\}^{l(n)}$ 不是 1-1 的。

2. 试证明，若单向置换存在，则存在伪随机发生器 G，使得映射 $G:\{0,1\}^n\rightarrow\{0,1\}^{l(n)}$ 是 1-1 的。

习题 14：设 G 是伪随机发生器，h 是多项式时间内可计算的置换（在相同长度的字符串上）。试证明，由 $G'(s)\stackrel{\text{def}}{=}h(G(s))$ 和 $G''(s)\stackrel{\text{def}}{=}G(h(s))$ 定义的 G' 和 G'' 都是伪随机发生器。

习题 15：设 G 是伪随机发生器，对它做如下修改：

1. 若 s 中 1 的数目恰好等于 $|s|/2$，则 $G'(s)\stackrel{\text{def}}{=}0^{|G(s)|}$；否则 $G'(s)\stackrel{\text{def}}{=}G(s)$。

2. 若 s 中 1 的数目恰好等于 $|s|/3$，则 $G''(s)\stackrel{\text{def}}{=}0^{|G(s)|}$；否则 $G''(s)\stackrel{\text{def}}{=}G(s)$。

试问哪一个是伪随机发生器？

习题 16：和第 2 章中的习题 9 类似，反驳如下猜想：

对每个伪随机发生器 G，函数 $G'(s)\stackrel{\text{def}}{=}G(s)\oplus s0^{|G(s)|-|s|}$ 也是伪随机发生器。

提示：设 g 是伪随机发生器，考虑定义在相同长度的字符串对上的 G，使得 $G(r,s)=(r,g(s))$。

习题 17：伪随机发生器的更为一般化的定义：以下定义源于标准定义，只是去掉了关于发生器的正则长度这一需求（即，不要求对所有 $|x|=|y|$ 都有 $|G(x)|=|G(y)|$）。一般化的伪随机发生器是满足以下两个条件的确定性多项式时间算法 G：

扩展性：对每个 $s\in\{0,1\}^*$，有 $|G(s)|>|s|$ 成立。

伪随机性（如定义 3.3.1）：总体 $\{G(U_n)\}_{n\in\mathbb{N}}$ 是伪随机的。

证明如下结论：

1. 若一般化的伪随机发生器存在，则标准的伪随机发生器也存在。

2. 设 G 是一个一般化的伪随机发生器，并令 $l:\mathbb{N}\rightarrow\mathbb{N}$，使得 $\{G(U_n)\}_{n\in\mathbb{N}}$ 和 $\{U_{l(n)}\}_{n\in\mathbb{N}}$ 是多项式时间不可分辨的。

(a) 试证明，$l(n)>n$ 只对有限多个 n 成立。

(b) 试证明，$G(U_n)$ 的长度不等于 $l(n)$ 的概率（对 n 来说）是可忽略不计的。

提示（有关 2b）：当 poly(n) 时间内不能由 n 求得 $l(n)$ 时，情况就变得比较困难了（否则，可以简单比较一下被检测的字符串和 $l(n)$ 的长度）。一般情况下，首先证明存在一个函数 $l':\mathbb{N}\rightarrow\mathbb{N}$，使得 $|G(U_n)|\neq l'(n)$ 的概率（对 n 来说）是可忽略不计的（提示：否则，$G(U_n)$ 的多项式乘积和 $U_{l(n)}$ 的多项式乘积是可区分的）。接着通过考虑一个识别器来证明 $l'(n)=l(n)$，当输入 1^n 和待检测的字符串 α 时，此识别器首先对 $G(U_n)$ 进行采样，然后将其长度和 $|\alpha|$ 相比较。

习题 18：对构造 3.3.2 做如下修改：用 $s_i\sigma_i=G_1(s_{i-1})$ 代替 $\sigma_i s_i=G_1(s_{i-1})$。试给出一种简单的方法来证明得到的算法也是伪随机的。

提示：不修改定理 3.3.3 的证明过程，而只修改 G_1 本身。

习题 19：扩展因子较大的伪随机发生器的另一种构造：设 G_1 是扩展因子为 $l(n)=n+1$ 的伪随机发生器，$p(\cdot)$ 是一个多项式。$G(s)$ 定义为 G_1 在 s 上迭代 $p(|s|)$ 次所得的结果（即，$G(s)\stackrel{\text{def}}{=}G_1^{p(|s|)}(s)$，其中 $G_1^0(s)\stackrel{\text{def}}{=}s$，$G_1^{i+1}\stackrel{\text{def}}{=}G_1(G_1^i(s))$）。

试证明，G 是伪随机发生器。

使用构造 3.3.2 的好处是什么？

习题 20：不可预测总体的另一种定义：对定义 3.3.6 做如下修改：只在那些不读取整个输入的（概率多项式时间）算法上量化。也就是说，每次执行这样的算法 A 时，若输入 $(1^{|x|}, x)$，A 最多读 x 中的 $|x|-1$ 比特。试证明，修改后的定义等价于原定义。

提示：修改后的定义的范畴小于原定义的范畴，所以我们只需证明如何把任意概率多项式时间算法 A 转化为这样一个算法：不读取整个输入且预测下一比特时仍然有相同的成功概率。这可以通过如下方法实现：假设不读取最后一个输入比特，并以此来逼近 A，因此无论何时 A 试图读最后一个输入比特，我们都将终止 A 并输出一个均匀选择的比特（否则，我们就如实地摸拟 A 了）。需要注意的是，当 A 读其最后一个输入比特时，（根据此时假设的 next_A 的定义；参见定义 3.3.6）它输出的预测比特的正确概率为 $\frac{1}{2}$。我们修改后的算法（即输出一个均匀选择的比特来猜测最后一个输入比特）就达到了这一成功概率。

习题 21：即时伪随机发生器：我们已经知道，不定长输出的伪随机发生器（参见 3.3.3 小节）是确定性的多项式时间程序，当给定随机种子时，它产生一个无限序列，使得该序列中每个多项式长的前缀都是伪随机的。即时伪随机发生器是不定长输出的伪随机发生器的一种特殊情况，即它保持并更新一种隐藏状态，使得能在种子的长度的多项式时间内产生下一输出比特，而不是当前产生的所有比特数的多项式。即时伪随机发生器是通过下一步函数来定义的。下一步函数把发生器的当前状态映射为一个由输出比特和下一状态组成的序对。即，设 g 是一个把长为 n 的字符串映射为长为 $(n+1)$ 的字符串的多项式时间算法，若对每个多项式 p，总体 $\{G_n^p\}_{n\in\mathbb{N}}$ 是伪随机的，其中 G_n^p 通过如下随机过程来定义：

均匀选择 $s_0\in\{0,1\}^n$。

从 $i=1$ 到 $p(n)$，重复如下步骤：$\sigma_i\cdot s_i\leftarrow g(s_{i-1})$，其中 $\sigma_i\in\{0,1\}$（且 $s_i\in\{0,1\}^n$）。

输出 $\sigma_1\sigma_2\cdots\sigma_{p(n)}$。

则称此算法 g 为即时伪随机发生器的下一步函数。

也就是说，s_0 是即时伪随机发生器的初始（随机）状态，s_i 是其输出 i 个比特后的状态（当然，这个总体的定义和构造 3.3.2 相似）。

1. 试证明，如果 G 是扩展因子为 $l(n)=n+1$ 的任意伪随机发生器，那么它也能构成某一即时伪随机发生器的下一步函数。

提示：利用前面提及的相似性。

2. 试证明，反过来不一定成立；即，如果 g 是即时伪随机发生器的下一步函数，那么它不一定是伪随机发生器。

提示：给定一个下一步函数 g'，对 $|r|=|s|$ 考虑下一步函数 $g(s\cdot r)=g'(s)\cdot 0^{|r|}$。

3. 还可证明，给定任何即时伪随机发生器（的下一步函数 g），可以很容易地构造出一个伪随机发生器。

提示：只需激活即时发生器足够多次。

即时伪随机发生器（的下一步函数）的定义可以保证发生器的当前状态的大小/长度不会随着产生的比特数的增加而增加。下面我们考虑一个条件稍微放宽的定义：允许当前状态的大小/长度适当地增长。例如考虑一个条件放宽的即时伪随机发生器的定义，即允许这样一个（多项式时间内可计算的）下一步函数 g，此函数把长为 m 的字符串映射成长为 $(m+O(1))$ 的

字符串。(所考虑的分布按如下方式重新定义:在$\{0,1\}^n$上均匀选择s_0,令$\sigma_i \cdot s_i = g(s_{i-1})$,其中$\sigma_i \in \{0,1\}$,然后输出$\sigma_1\sigma_2 \ldots \sigma_{p(n)}$;但是这里$|s_{p(n)}| \neq |s_0|$,除非和前文一样有$|g(s)| = |s|+1$。)

- 试证明,使用即时伪随机发生器的这一条件放宽的定义不能保证每个输出比特都在种子的长度的多项式时间内产生(即不考虑当前所产生的比特数)。
- 试证明前面 3 项仍然成立。
- 设g是条件放宽的即时伪随机发生器的下一步函数,$T_g(m)$表示输入的长为m时计算g的复杂度。试给出产生$t(n)$比特的复杂度的上界,其中这$t(n)$比特是在基于g的条件放宽的即时伪随机发生器中由长为n的种子得到的。比较这个界和从条件未放宽的即时伪随机发生器得到的界。
- 我们允许当前状态每步增大多少,才能使输出多项式多个比特时仍能保持多项式时间操作?

习题 22:哈希族的构造:把l维二进制向量与长为l的字符串相结合。

1. 考虑把长为n的字符串映射为长为m的字符串的函数集S_n^m。S_n^m中的一个函数$h_{A,b}$用一个序对(A,b)表示,其中A是$m \times n$的二进制矩阵,b是一个m维二进制向量。n维二进制向量x被函数$h_{A,b}$映射成m维二进制向量(此向量是A乘x得到的),再把向量b和得到的向量相加(即$h_{A,b}(x) = xA + b$)。试证明,这样定义的S_n^m组成了一个(如 3.5.1.1 节所述的)哈希族。

2. 当把$m \times n$矩阵组限定为 Toeplitz 矩阵组时,重复第一项。$m \times n$的 *Toeplitz* 矩阵组对所有i,j都满足$T = \{T_{i,j}\}$,$T_{i,j} = T_{i+1,j+1}$。

需要注意的是,二进制$m \times n$ Toeplitz 矩阵可以用长为$n+m-1$的字符串来表示,而表示任意$m \times n$的二进制矩阵则需要长为$n \cdot m$的字符串。

提示:对每个$\{0,1\}^n$中的$x \neq x'$和每个$v, v' \in \{0,1\}^m$,证明满足$h(x) = v$和$h(x') = v'$的函数$h \in S_n^m$的个数是独立于v和v'的。例如,在第一部分中每个这样的函数都和序对(A,b)相关联。我们考虑这样的序对,满足等式$xA + b = v$和$x'A + b = v'$的系统(或者等价地,$xA + b = v$和$(x - x')A = v - v'$),其中,x, x', v和v'是固定的,A和b的项是未知的。

习题 23:哈希族的另一种构造:这里我们利用了一种在有限域 GF(2^n)上的可有效处理的表达式。这就需要一个在二元域 GF(2)上的度为n的不可约多项式。对特定的n值,存在一个好的表达式:对$n = 2 \cdot 3^e$(e是整数),多项式$x^n + x^{n/2} + 1$在 GF(2)上是不可约的(153, Thm. 1.1.28)。

设$m \leq n$,考虑如下函数集S_n^m(该函数把长为n的字符串映射为长为m的字符串)。S_n^m中的函数$h_{a,b}$可以用两个元素$a, b \in$ GF(2^n)来表示,并且对每一个$x \in$ GF(2^n),$h_{a,b}(x)$的值等于$ax + b$的长为n的表达式的前m比特,其中算法作用于域 GF(2^n)上。

1. 试证明,这样定义的S_n^m构成了哈希族。
2. 试证明,S_n^m中正则的函数所占的比例是按指数衰减的(即,2^{n-m}到 1)。

提示:对第一部分,可以利用这一事实:对每个$x \neq x'$和每个$v, v' \in$ GF(2^n),存在单个序对(a,b),使得$ax + b = v$,$ax' + b = v'$。第二部分使用这一事实:对每个$a \neq 0$和b,映射$x \to ax + b$是 1-1。

习题 24:另一哈希引理:设$m, n, S_n^m, b, X_n, \delta$如引理 3.5.1 中所述。试证明,对每个集合

$S\subseteq\{0,1\}^m$，S_n^m中最多只有 $2^{(b-m-\log_2|S|)}\cdot\delta^{-2}$的 h 满足

$$\Pr[h(X_n)\in S]\in(1\pm\delta)\cdot\frac{|S|}{2^m}$$

提示：按照引理 3.5.3 的证明，若 $h(x)\in S$，则定义 $\zeta_x(h)=1$，否则为 0。

习题 25：又一个哈希引理：设 m、n 和 S_n^m 如前所述，并令 $B\subseteq\{0,1\}^n$ 和 $S\subseteq\{0,1\}^m$ 是集合。试证明，S_n^m 中最多只有$\frac{2^m}{|B|\cdot|S|}\cdot\delta^{-2}$的 h 满足

$$|\{x\in B: h(x)\in S)\}|\in(1\pm\delta)\cdot\frac{|S|}{2^m}\cdot|B|$$

提示：定义一个在 B 上均匀分布的随机变量 X_n。

习题 26：一种失败的伪随机函数的构造：构造一个函数总体，其中 F_n 中的函数定义如下。对每个 $s\in\{0,1\}^n$，函数 f_s 定义为：

$$f_s(x)\stackrel{\text{def}}{=}G_{\sigma_n}(\cdots(G_{\sigma_2}(G_{\sigma_1}(x)))\cdots)$$

其中 $s=\sigma_1\cdots\sigma_n$ 和G_σ 如构造 3.6.5 中所述。即，x 和 s 在构造 3.6.5 中的角色相互转换(即，树根用 x 标记，f_s 作用于 x 的值可以沿着索引 s 所对应的路径得到)。试证明，产生的函数总体不一定是伪随机的(尽管 G 是伪随机发生器)。

提示：首先证明如果伪随机函数存在，那么存在一个伪随机发生器 G 满足 $G(0^n)=0^{2n}$。

习题 27：可直接访问的伪随机发生器：可直接访问的伪随机发生器是一种确定性的多项式时间算法 G。用此算法任何概率多项式时间预言机都不能区分以下两种情况：

1. 预言机的新质疑可以用硬币的独立投掷来响应(重复两次相同的质疑会产生同样的响应)。

2. 首先均匀地选择一个长为 n 的随机“种子”s，然后每个质疑 q 由$G(s,q)$响应。

$G(s,i)$这一比特可以视为与种子 s 相对应的序列的第 i 位，其中 i 用二进制表示。

- 试证明，(普通)伪随机发生器的存在性隐含了可直接访问的伪随机发生器的存在性。

提示：可直接访问的伪随机发生器实质上是一个伪随机函数总体。

- 试证明，若修改此定义，使之只允许一元质疑，则将产生另一种条件放宽的即时伪随机发生器的定义(如习题 21 中的定义)。

提示：给定一个条件放宽的即时伪随机发生器的下一步函数 g，我们可以得到一个发生器 G，它令 $G(s,1^i)$为条件放宽的即时伪随机发生器在原始状态 s 下产生的第 i 个比特，从而能够支持“直接访问”一个长为多项式的序列。相反地，给定这样一个“一元可直接访问”的机器 G，我们通过令 $g(s,1^i)=(G(s,1^{i+1}),(s,1^{i+1}))$可以获得一个下一步函数 g(即第 i 个状态是$(s,1^i)$，其中 $s\equiv(s,\lambda)$是有限状态)。

- 试评估可直接访问的伪随机发生器凌驾于即时伪随机发生器之上的优点，即使是在只能直接访问多项式长度的伪随机序列的情况下也是比后者优秀的。

习题 28：下面考虑如定义 3.6.4 所定义的伪随机函数总体的长度函数 $l:\mathbb{N}\to\mathbb{N}$。

1. 试说明，若 $l(n)>\log_2 n$，这样的伪随机函数就能生成一个伪随机发生器。其实只要$l(n)^{l(n)}>n$就足够了。

2. 如果 $l(n)^{l(n)}\leq n$，那么不用基于任何假设就能给出一个长为 $l:\mathbb{N}\to\mathbb{N}$ 的伪随机函数总体的构造。

习题 29：设$\{f'_s:\{0,1\}^{d(|s|)}\to\{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$是一个一般化的伪随机函数总体。假设

G'是多项式时间内可计算的，总体$\{G'(U_n)\}_{n\in\mathbb{N}}$像定义 3.2.8 定义的那样是伪随机的（如果 G' 是一个伪随机发生器，那么它就同时满足这两个条件，但是反过来并不成立）。

试证明，由 $f_s(x) \stackrel{\text{def}}{=} G'(f's(x))$ 所定义的一般化函数总体 $\{f_s: \{0,1\}^{d(|s|)} \to \{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$ 是伪随机的。

提示：参考定理 3.6.11 的证明。

习题 30：加快伪随机函数的构造速度（Leonid Levin 提出的）：对某个 $d,r:\mathbb{N}\to\mathbb{N}$，像定义 3.6.9 中那样考虑一个一般化的伪随机函数总体

$$F \stackrel{\text{def}}{=} \{f_s: \{0,1\}^{d(|s|)} \to \{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$$

Primes_m 表示在区间$(2^{m-1},2^m)$上的质数的集合。对任意 $d':\mathbb{N}\to\mathbb{N}$，考虑一个新的函数总体，

$$F' \stackrel{\text{def}}{=} \{f'_{s,p}: \{0,1\}^{d'(|s|)} \to \{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*,\,p\in \text{Primes}_{d(|s|)}}$$

其中 $f'_{s,p}(x) \stackrel{\text{def}}{=} f_s(x \bmod p)$，$\{0,1\}^{d'(|s|)}$和$\{0,1\}^{d(|s|)}$分别与$\{0,\ldots,2^{d'(|s|)}-1\}$和$\{0,\ldots,2^{d(|s|)}-1\}$相关联。

其关键在于，F' 中的函数在与 F 的时间复杂度相关的时间内是可计算的。无论何时 $d'(n)\gg d(n)$（例如，$d'(n)=n^2$，$d(n)=\log_2^2 n$），（若与构造 3.6.10 相比）它都会在 F'的时间复杂度上产生加速。

1. 试证明，若 $d(n)=\omega(\log n)$，则 F'是伪随机的。

2. 试说明，另一方面，若 $d(n)=O(\log n)$（且 $d'(n)>d(n)$），则 F'不是伪随机的。

需要注意的是，一般说来，F'的“伪随机性”（因为可以证明 F'不是随机的足够长的运行时间是可以量化的）是依赖于 $d:\mathbb{N}\to\mathbb{N}$ 的。特别地，可以在 d 的指数的时间内证实 F'不是随机的。

提示（第 2 部分）：检查所有可能的 p，试着证明目标函数确实使用了以 p 为模的变形（提示：对选定的 p，任意两个不同的 $x,y\in\{0,1\}^{d'(|s|)}$满足 $x\equiv y \pmod p$即可证明）。

提示（第 1 部分）：考虑把上述构造作用于均匀函数总体 H，而非伪随机总体 F。主要问题在于证明得到的总体 H'是伪随机的（F'和 H'是不可分辨的，否则我们就能分辨 F 和 H）。

提示（第 1 部分，附加提示）：我们所指的是这样的函数总体 $H'=\{H'_n\}_{n\in\mathbb{N}}$，其中 H'_n 如下定义：均匀选择一个函数 $h:\{0,1\}^{d(n)}\to\{0,1\}^{r(n)}$，且 $p\in \text{Prime}_{d(n)}$，并令 $H'_n=h'_p$，使得 $h'_p(x)=h(x \bmod p)$。如果不同的质疑 $x_1,\ldots,x_t\in\{0,1\}^{d'(n)}$有不同的模 p 剩余类，那么来自 h'_p 的响应是在$\{0,1\}^{r(n)}$上独立且均匀分布的。于是我们首先必须给出均匀选择的 $p\in\text{Prime}_{d(n)}$的上一结果的概率下界。我们可以限定互补结果的概率的上界（即 $\exists i\neq j$，使得 $x_i=x_j \pmod p$）。对不同的 $x,y\in\{0,1\}^{d'(n)}$，$x\equiv y\pmod p$成立当且仅当 p 整除 $x-y$。这里，证明过程可以利用 p 是素数这一事实来简化；[①]均匀选择的长为 $d(n)$的素数整除长为 $d'(n)$的整数的概率最多是$\frac{d'(n)/d(n)}{|\text{Prime}_{d(n)}|}$，它是 $\Theta(d'(n)\cdot 2^{-d(n)})$。

习题 31：习题 30 的另一种构造：设 F 和 d'如习题 30 所述，并令 $s_{d'(n)}^{d(n)}$是一个哈希族（如 3.5.1.1 节中所定义）。对每个 $s\in\{0,1\}^*$ 和 $h\in s_{d'(|s|)}^{d(|s|)}$，定义 $f'_{s,h}$，使得 $f'_{s,h}(x)=f_s(h(x))$，并设

① 如何修改构造使得 p 在$\{2^{d(n)-1},\ldots,2^{d(n)}-1\}$中的所有整数上均匀选择？

$$F' \stackrel{\text{def}}{=} \{f'_{s,h}: \{0,1\}^{d'(|s|)} \to \{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*, h\in S_{d'(|s|)}^{d(|s|)}}$$

（虽然这个构造需要的种子比习题 30 中的更长，但是可以利用更小的接近期望特性的函数族。）

1. 试证明，若 $d(n)=\omega(\log n)$，则 F' 是伪随机的。

2. 另一方面，试证明，若 $d(n)=O(\log n)$ 且 $r(n)>d(n)$，则 F' 不是伪随机的。

提示(第 2 部分)：对任何不同的 $x,y\in\{0,1\}^{d'(n)}$，以及把长为 $d'(n)$ 的字符串映射成长为 $r(n)$ 的字符串的均匀选择的函数，x 和 y 映射成同一个像的概率是 $2^{-r(n)}$。然而，在均匀选择的 $f'_{s,h}$ 的作用下，x 和 y 映射成同一个像的概率不低于 $\Pr[h(x)=h(y)]=2^{-d(n)}$。

习题 32：伪随机函数的另一个定义：为了简化，本习题按布尔函数总体的方式描述（和定义 3.6.9 类似，有 $d(n)=n$ 和 $r(n)=1$）。也就是说，我们考虑一个布尔函数总体 $\{f_s: \{0,1\}^{|s|}\to\{0,1\}\}_{s\in\{0,1\}^*}$，并令 F_n 在多重集合 $\{f_s\}_{s\in\{0,1\}^n}$ 上均匀分布。如果对每个概率多项式时间预言机 M、每个多项式 $p(\cdot)$ 和所有足够大的 n，有

$$\Pr[\mathrm{corr}^{F_n}(M^{F_n}(1^n))] < \frac{1}{2}+\frac{1}{p(n)}$$

那么我们就说函数总体 $\{F_n\}_{n\in\mathbb{N}}$ 是不可预测的。

其中，$M^{F_n}(1^n)$ 假设为形如 $(x,\sigma)\in\{0,1\}^n\times\{0,1\}$ 的值，使得 x 不是出现在计算 $M^{F_n}(1^n)$ 中的质疑，$\mathrm{corr}^f(x,\sigma)$ 定义为断言"$f(x)=\sigma$"。直观上，得到 f 在其选择的点上的值后，机器 M 输出一个新的点（即 x），同时输出猜想的在这个点上的 f 的值（即 σ）。$\mathrm{corr}^f(x,\sigma)$ 的值表示此猜想中 M 是否正确。

若 $F=\{F_n\}_{n\in\mathbb{N}}$ 是可以有效计算的，试证明，F 是伪随机的当且仅当 F 是不可预测的。

提示：证明过程与定理 3.3.7 类似。

习题 33：伪随机函数一种错误的定义：像习题 32 那样我们再次考虑布尔函数的总体，考虑如下具有弱不可预测性的函数总体的定义：预言机 M 把均匀选择的 $x\in\{0,1\}^n$ 作为输人，然后质疑它选择的多项式多（比 x 多）个点上 f 的预言，则将输出对 $f(x)$ 的猜测。

我们要求对每个概率多项式时间预言机 M（对它自己的输人不质疑预言）、每个多项式 $p(\cdot)$ 以及所有足够大的 n，有

$$\Pr[M^{F_n}(U_n) = F_n(U_n)] < \frac{1}{2}+\frac{1}{p(n)}$$

也就是说，不像习题 32 的公式那样，预言机不能选择在哪个点上必须预测函数值（但是这个点是随机的并作为输入给出）。

1. 试证明，任何伪随机函数总体都是弱不可预测的。

2. 假设伪随机函数总体存在，证明弱不可预测函数总体存在，但不是伪随机的。

本习题与有缺陷的结论（它出现在本书手稿的早期版本中）相矛盾。这个缺陷是 Omer Reingold 指出的。

提示：对第 1 部分，试证明如习题 32 所定义的不可预测性隐含了弱不可预测性。或者（像习题 32 那样）直接证明。对第 2 部分，修改一个伪随机函数总体，使得在 F_n 的值域上的每个 f 都满足 $f(0^n)=0$。

习题 34：试图强化函数的弱不可预测性的概念使之等价于伪随机函数，但这是不成功的：作为习题 33 的后续，假设我们通过如下方式强化这一条件：允许从任意多项式时间内可构造

的总体中选择一个输入 x。即，如果对每个概率多项式时间预言机 M（对它自己的输入不质疑预言），每个多项式时间内可构造的总体 $\{X_n\}_{n\in\mathbb{N}}$，其中 X_n 在 $\{0,1\}^n$ 上取值，对每个多项式 $p(\cdot)$ 和所有足够大的 n，有

$$\Pr[M^{F_n}(X_n) = F_n(X_n)] < \frac{1}{2} + \frac{1}{p(n)}$$

我们就说函数总体 $F=\{F_n\}_{n\in\mathbb{N}}$ 是第 2 种弱不可预测的。

同样，试证明此定义是伪随机总体的必要非充分条件。

提示：修改函数总体，使得在 F_n 的值域上的每个 f 满足 $f(f(a^1)f(a^2)\ldots f(a^n))=0$，其中 $a^1,\ldots,a^n\in\{0,1\}^n$ 是一些易于计算的字符串（例如，$a^i=0^{i-1}10^{n-i}$）。

习题 35：令 $t:\mathbb{N}\to\mathbb{N}$，使得输入 n 时可以在 poly(n)时间内计算 $t(n)$。令 $\{F_n\}_{n\in\mathbb{N}}$ 和 $\{H_n\}_{n\in\mathbb{N}}$ 是任何概率多项式时间预言机都不能区分的两个函数总体。试证明，置换总体 $\{\mathrm{DES}^{t(n)}_{F_n}\}_{n\in\mathbb{N}}$ 和 $\{\mathrm{DES}^{t(n)}_{F_n}\}_{n\in\mathbb{N}}$（如 3.7.2 小节所定义）是任何概率多项式时间预言机都不能区分的。而且，当此预言机给定对此置换及其逆置换的访问时（如定义 3.7.5），它也成立。

提示：可以利用混合证明法把 F_n 的 $t(n)$ 个独立副本和 H_n 的 $t(n)$ 个独立副本联系起来。其中第 i 个混合是 $\mathrm{DES}_{F_n^{(t(n))},\ldots,F_n^{(i+1)},H_n^{(i)},\ldots,H_n^{(1)}}$。需要注意的是，对置换 $\mathrm{DES}_{F_n^{(t(n))},\ldots,F_n^{(i+2)},g,H_n^{(i)},\ldots,H_n^{(1)}}$（和它的逆置换一样）的预言访问可以利用对函数 g 的预言访问来近似。

习题 36：令 F_n 和 $\mathrm{DES}^t_{F_n}$ 如构造 3.7.6 中所述，无论选择怎样的总体 $F=\{F_n\}_{n\in\mathbb{N}}$，总体 $\mathrm{DES}^2_{F_n}$ 都不是伪随机的。

提示：首先证明总体 $\mathrm{DES}^1_{F_n}$ 不是伪随机的（一个质疑就足够了）。为了把 $\mathrm{DES}^2_{F_n}$ 和随机置换区分开来，使用两个相关的质疑。

习题 37（Luca Trevisan 提出的）：假设伪随机函数总体存在，试证明存在一个不严格的伪随机置换总体。

提示：首先构造一个伪随机构造总体，它的种子的长度小于或等于域大小的对数。接着，修改它使得这个种子被映射为一个固定的点（例如，全零字符串），这样修改后的总体仍然是一个置换。

习题 38：和习题 36 相似，证明 $\mathrm{DES}^3_{F_n}$ 不是严格伪随机的。

提示：需要更深入的思考和不止一对质疑。一定要对两个预言都质疑。

第 4 章

零知识证明系统

在本章中，我们讨论零知识(ZK)证明系统。不严格地说，这样的证明系统具有使对方确信并且不泄漏信息(不泄漏除证明论点有效性所必需的证据之外的任何信息)的良好特性。换句话说，接收到某个结论的零知识证明等价于被一个信任机构告知结论成立(参见图 4.1 中的描述)。本章的主要结果是给出一种产生 ***NP*** 系统中所有语言的零知识证明系统的方法。这种方法可以用任何一种比特承诺方案来实现，而所使用的比特承诺方案可以用任何伪随机发生器产生。这种方法的重要性在于它的通用性，这对于许多应用都是关键的。特别地，几乎人们在实践中想要证明的所有论断都可以表述为是否是 ***NP*** 中成员的声明。另外，我们还进一步讨论了零知识概念的更为高级的方面，以及它们对零知识概念应用的影响。

结构安排　4.1～4.4 节介绍基础知识。特别地，我们从介绍零知识概念提出的动机(4.1 节)开始，然后定义交互证明(4.2 节)和零知识(4.3 节)的概念并给出一些相应的例子，最后我们给出一种 ***NP*** 系统中每种语言都适用的零知识证明系统(4.4 节)。接下来介绍一些高级主题(参见图 4.2)。除非(在下面的列表和图 4.3 中)特别声明，这些高级主题中的每一个都可以独立进行阅读。

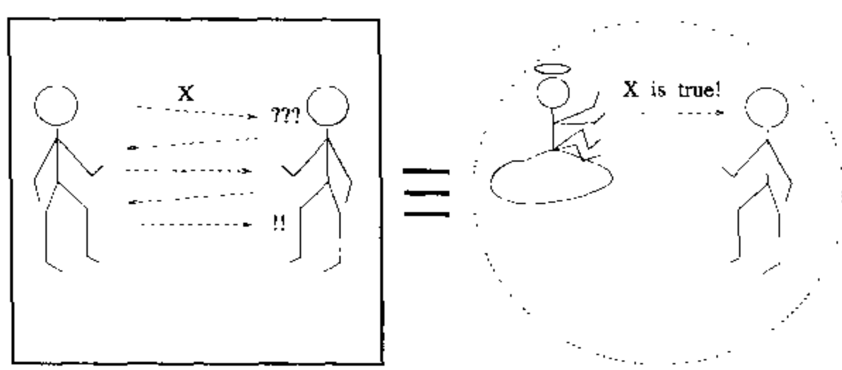


图 4.1

- 在 4.5 节中，我们给出了零知识证明的一些否定结果。这些结果揭示了 4.4 节中结果的“最优性”，从而导致了 4.6 节和 4.8 节中零知识证明衍生系统的产生。
- 在 4.6 节中，我们给出了零知识的一个主要的弱化系统，并证明它在并行组合运算下是封闭的(这个结论对一般的零知识是不成立的)。这里，我们引出一个叫做证据不可分辨性的概念，这个概念与证据隐藏性有关(也定义并讨论了这个概念)。

- 在 4.7 节中，我们定义和讨论了知识的（零知识）证明。
- 在 4.8 节中，我们讨论了交互证明的一个弱化系统，叫做计算合理证明（或理论）。

Section 4.5:	*Negative results*
Section 4.6:	*Witness indistinguishability* and *witness hiding*
Section 4.7:	*Proofs of knowledge*
Section 4.8:	*Computationally sound proofs (arguments)*
Section 4.9:	*Constant-round* zero-knowledge systems
Section 4.10:	*Non-interactive zero-knowledge* proofs
Section 4.11:	*Multi-prover* zero-knowledge proofs

图 4.2

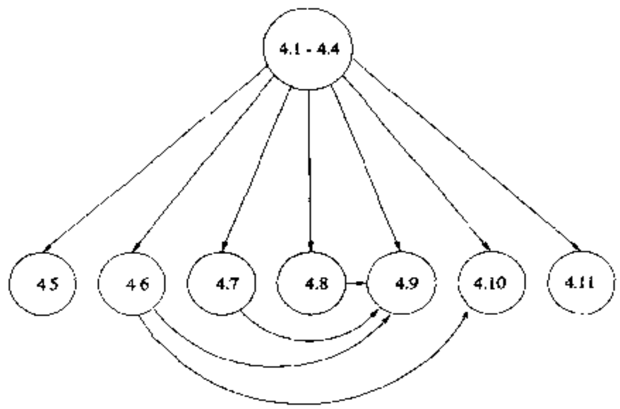


图 4.3

- 在 4.9 节中，我们给出了常数轮零知识系统的两个构造。第一个是一个交互证明系统，第二个是一个争论系统。4.8.2 小节（讨论完备隐藏承诺方案）是第一种构造的预备知识，而 4.8、4.7、4.6 节是第二种构造的预备知识。
- 在 4.10 节中，我们讨论了非交互零知识证明。证据不可分辨性的概念（在 4.6 节中定义）是 4.10.3.1 节中给出的结果的预备知识。
- 在 4.11 节中，我们讨论了多证明者证明系统。

同往常一样，我们在最后一节（4.12 节）给出总结。

教学提示　图非同构的交互证明系统（在 4.2 节中给出）和图同构的零知识证明（在 4.3 节中给出）仅仅是描述性的例子。因此，应该避免分析这些例子的细节。

4.1　零知识证明：动机

典型的“密码学”问题包括如何在互不信任的多方之间透露（事先确定的）“信息片断”。也就是说，每个参与方都拥有秘密，并且它们都想透露这些秘密的一部分。这些秘密是由一些大家都知道的信息完全决定的或部分决定的，因此说透露秘密的正确值是有意义的。问题就在

于如何验证最近透漏的秘密的部分，而不泄漏秘密的其他部分。为了澄清问题，让我们看一个特定的例子。

假定一个系统中的所有用户都在一个可以公开访问的存储介质上保存他们用各自的密钥加密的他们所有文件系统的备份。又假定某一时刻，一个用户（Alice）想要将她的一个文件（存放在她的备份中）里某条记录的明文提供给另一个用户（Bob）。一个显然的“解决方法”是Alice简单地将（明文）记录发送给Bob。这种“解决方法”的问题是Bob没有办法知道Alice是真的将真正的记录（在Alice的公共备份中加密的那条）发送给他，还是从其他地方随便给他发了一个记录。通过将她的密钥出示给Bob，Alice可以很简单地证明她发送给Bob的记录是正确的。然而，这样做将会使Alice的所有文件的内容都泄漏给Bob，Alice当然不希望这种事情发生。问题是Alice能否使Bob确信她确实将正确的记录透漏给了Bob而又不会泄漏其他“知识。”

一个类似的问题可以正式地描述如下。令 f 是一个单向置换，b 是关于 f 的一个核心断言。假设一方 A 有一个字符串 x，而另一方记做 B，只有 $f(x)$。而且，假定 A 想要将 $b(x)$ 透漏给 B，而又不泄漏其他任何信息。显然的“方法”是让 A 把 $b(x)$ 发给 B，但是，正如上面解释过的一样，B 没有办法知道 A 是否真的发送了正确的比特（而不是相反的比特）。A 实际上可以通过将 x 也发送给 B 来证明他发送了正确的比特（即 $b(x)$），但这违背了 A 的初衷。同样地，这个问题也归结为 A 如何向 B 证明他给了正确的比特（即 $b(x)$），同时又不泄漏额外的“知识”。

总之，上面的问题就是证明一个论断而又不泄漏其他任何信息是否是可能的。这样的证明只要存在，就叫做零知识，并且在“密码”协议的设计中扮演着中心角色。

不严格地说，零知识证明是在提供证据的同时又不泄漏任何信息（即“无知识”）的证明。在本节的其余部分，我们讨论“证明”的概念和“不泄漏多余信息（即无知识）”可能的意思。

4.1.1 证明的概念

证明就是使我确信某些事情的任何东西。

Shimon Even 在图算法课上回答他的学生的话(1978)

我们讨论证明的概念主要是想揭示它的本质。

4.1.1.1 静态对象与交互过程

通常在数学上，“证明”是一个包含许多论断的固定过程，这些论断是自明的或可以通过自明规则从前面的论断中推导出来。实际上，用术语“被共同接受”来替换术语“自明”更准确一些。事实上，在证明的正式研究（即逻辑学）中，被共同接受的论断叫做公理，而被共同接受的规则叫做推导规则。我们要强调数学证明的两个特点：

1. 证明被看作是固定的对象；
2. 证明与它们的结果（即定理）同样都是基本的。

然而，在人类活动的其他领域，“证明”的概念有更广泛的含义。特别地，证明不是一个固定的对象，而是一个过程，通过这个过程，断言的有效性被确立起来。例如，对证人在法庭上的交互询问是一种法律上的证明，而不能给出对手问题的足够答案可能是一种哲学上、政治上、

有时甚至是技术讨论中的证明。另外，在现实生活中，一般都认为证明相对于它的结果是处于从属地位的（从重要性上说）。

总之，在“经典”的数学中，证明具有静态的特性（例如，它们都是“写出来的”），而在现实生活中，证明具有动态的特性（即，它们都是通过交互建立起来的）。证明概念的动态解释在我们的环境中更加适合，在我们的环境中证明用作“密码学”协议中的工具（即子协议）。而且，证明概念的动态解释（至少在一定意义上）对零知识证明概念的非平凡性是至关重要的。

4.1.1.2　证明者和验证者

无论是在数学场合下还是在现实生活中，证明者的概念都是不证自明的：证明者是提供证明的（有可能是隐含的也有可能是先验的）实体。反之，验证者的概念在这些讨论中要更明确一些，在这些讨论中重点强调的是验证过程，换句话说就是验证者扮演的角色。无论是在数学上还是在现实生活中，证明都是通过验证过程来定义的。验证过程一般认为相对简单，而重点是提供证明的当事者/人（即证明者）。

验证任务的复杂性与定理证明任务的复杂性之间的不对称性可以通过复杂类 ***NP*** 刻划出来，***NP*** 类可以看作是一类证明系统。每种语言 $L\in \mathcal{NP}$ 都有一个对于形如“$x\in L$”的结论证明的有效验证过程。注意到，每个 $L\in \mathcal{NP}$ 都可以用一个多项式时间可识别的关系 R_L 来刻划，使得

$$L=\{x: \exists y \text{ s.t. } (x,y)\in R_L\}$$

并且仅当 $|y|\leq \text{poly}(|x|)$ 时 $(x,y)\in R_L$。

这样，对于“$x\in L$”形式的成员声明的验证过程就包括将识别 R_L 的（多项式时间）算法应用于声明（记做）x 和预期的证明，记做 y。任意满足 $(x,y)\in R_L$ 的 y 都被认为是成员 $x\in L$ 的一个证明。因此，正确的论断（即 $x\in L$）而且只有正确的论断在这个证明系统中才有证明。注意，验证过程是“容易的”（即是多项式时间的），而提出证明可能会“很难”（如果实际上 ***NP*** 不包含在 ***BPP*** 中）。

值得注意的是，在任何证明系统中都存在对证明者的“不信任态度”。如果验证者相信证明者，那么证明就不需要了。因此，当讨论证明系统的时候，都假定验证者不信任证明者，对证明者所说的都持怀疑态度。

4.1.1.3　完备性和合理性

证明系统（即验证过程）的两个基本特性是它的*合理性*（或*有效性*）和*完备性*。合理性是说验证过程不能“误导”接受错误的论断。换句话说，合理性揭示了验证者具有保护自己不接受错误论断的能力（不论证明者采用怎样的方式欺骗它）。另一方面，完备性揭示了证明者具有能够使验证者相信正确论断（属于某些预先确定的正确结论集）的能力。注意，这两个特性对于证明系统概念都是实质性的。

我们在这里指出，并不是每一个正确论断集都有一个“合理的”证明系统，在这个证明系统中正确论断都能够被证明（而错误论断都不能被“证明”）。这个基本的事实在如哥德尔的不完备定理和图灵的终止问题的不确定性定理等结论中有更精确的阐述。我们强调，在本章中我们将研究范围限定在具有“有效证明系统”的（正确论断）集合类中。实际上，4.2 节将讨论并给出“有效证明系统”的概念和公式化描述。我们提前指出，证明系统的有效性与它的验证过

程的有效性息息相关。

4.1.2　获得知识

我们已经指出，零知识证明就是验证者不能获得（在有效性论证之外的）“知识”的证明。读者可能想知道知识到底是什么，知识的获得又是什么。在讨论零知识证明时，我们回避第一个问题（这个问题太复杂了），而直接处理第二个问题。即，我们不给出知识的定义，而是给出一个可以非常公正地说明没有获得知识的通用例子。幸运的是，这样的方法对于密码学所关心的问题已经足够了。

为提出零知识的定义，我们先考虑 Alice 和 Bob 双方之间的一个对话。首先假设这个对话是单向的，特别地，在这个对话中 Alice 只说，Bob 只听。显然，我们可以说 Alice 从对话中没有获得任何信息。另一方面，Bob 从对话中可能获得，也可能没有获得知识（取决于 Alice 说的内容）。例如，如果 Alice 说的只是“1＋1＝2”，那么 Bob 从对话中什么知识也得不到，因为这个事实他本来就知道。但是，如果 Alice 透漏给 Bob 的是一个关于 $\boldsymbol{P}\neq\boldsymbol{NP}$ 的证明，那么显然 Bob 从对话中获得了知识。

为了给出更准确的定义，我们现在考虑 Alice 和 Bob 的对话，在这个对话中 Bob 会问 Alice 一些关于一个比较大的（双方都知道的）图的问题。先考虑 Bob 问 Alice 这个图是否是欧拉图的情形。很明显，Bob 从 Alice 的回答中得不到任何知识，因为 Bob 自己可以很容易地确定这个问题的答案（通过运行线性时间的判别程序[1]）。另一方面，如果 Bob 问 Alice 这张图是否是哈密尔顿图，那么不管 Alice 怎么回答这个问题，我们都不能说 Bob 从中没有获得知识（因为我们不知道是否存在这样一个有效程序：通过这个程序，Bob 能自己确定答案，在假定 $\boldsymbol{P}\neq\boldsymbol{NP}$ 的情况下，这样的有效程序是不存在的）。因此，如果 Bob 关于这张公开图的计算能力有所增加的话，那么我们就说 Bob 从这个对话中获得了知识（即，如果在对话后他能够很容易地计算他在对话前很难计算的问题）。另一方面，如果 Bob 在对话前后都能计算关于这个图的某些东西，我们就说 Bob 从这个对话中没有获得任何知识。也就是说，只有当 Bob 收到一些他没有能力算出的结果，才能说 Bob 获得了知识。到目前为止，Alice 如何产生这个计算上不可行的运算（例如，告诉 Bob 这张图是否是哈密尔顿图）的问题一直没有考虑。回到刚才，我们注意到 Alice 可能有一个初步的轮廓或者可能拥有一些另外的线索，这使得她有能力产生这样的运算，反之则不能（特别地，对 Bob 来说不能完成运算是因为他没有这些线索）。

知识与信息

我们要强调的是（这里讨论的）知识不同于（信息论意义上的）信息。主要的两点区别是：

1. 知识是与计算难度相关的，而信息则不是。在前面提到的例子中，Alice 回答形如“这张图是欧拉图吗”的问题的例子和 Alice 回答形如“这张图是哈密尔顿图吗”的问题的例子中透漏的知识是不同的。从信息论的角度看，这两个例子并无区别（即，两个例子中答案都由问题惟一确定，因此 Bob 都没有获得信息）。

---

[1] 例如，通过应用欧拉定理来判断。欧拉定理断言一个图是欧拉图的充要条件是：图是连通的并且所有顶点都是偶数的度。

2. 知识主要与大家都知道的对象有关，而信息主要与那些只有部分信息大家都知道的对象有关。考虑这个例子，Alice 通过投掷硬币然后告诉 Bob 结果的方式来回答上面的每个问题。从信息论的角度看，Bob 从 Alice 身上获得与事件相关的信息。但是，我们说 Bob 没有从 Alice 那里获得任何知识，因为他也可以自己来投掷硬币。

4.2 交互证明系统

在这一节中，我们介绍交互证明系统的概念，并给出一个非平凡交互证明系统的例子（即形如“下面的两个图是不同构的”的论断）。这个例子就是为了引入零知识交互证明而列举的。交互证明系统本身就是很有趣的，并且具有重要的复杂性理论应用[①]。

4.2.1 定义

交互证明系统的定义特指与证明系统相关的两个计算任务：“产生”一个证明和验证证明的有效性。这两个计算任务分别由两个具有交互关系的参与方（即证明者和验证者）来执行。在某些情况下，交互可以非常简单，特殊情况下，可以是单向的（即，证明者给验证者发送一个称为“证明”的文本）。一般来说，交互可能很复杂，可能采取验证者询问证明者的形式。我们通过定义这样的询问过程来开始。

4.2.1.1 交互

两个参与方之间交互的定义是很显然的。惟一值得注意的一点是，交互通过公共输入（对两个参与方而言）进行参数化。在交互证明系统上下文中，公共输入表示需要被证明的论断。我们先定义交互机的概念，然后定义两个交互机交互的概念。读者可以直接跳到 4.2.1.2 节，在这一节中给出了一些（关于交互机的）重要约定，很全面（如果有遗漏，也不会很多）。

定义 4.2.1（交互机）：

- **交互图灵机（ITM）**是（确定的）多磁带图灵机。这些磁带包括一个只读输入磁带、一个只读随机磁带、一个读写工作磁带、一个只写输出磁带、一对通信磁带和一个只包含一个单元的读写交换磁带。通信磁带对中的一个是只读的，而另一个是只写的。
- 每个 **ITM** 都与一个比特 $\sigma\in\{0,1\}$ 相关联，这个比特称作 **ITM** 的标识。在一种配置下，如果 **ITM** 交换磁带的内容等于机器标识，就称它为活动的。否则，该 **ITM** 就称是空闲的。当 **ITM** 空闲的时候，机器的状态、机器头在不同磁带上的位置以及 **ITM** 可写磁带上的内容都是不能修改的。
- 输入磁带上的内容叫做**输入**，随机磁带上的内容叫做**随机输入**，终端输出磁带上的内容叫做**输出**。在机器的活动期间，写在只写通信磁带上的内容称为在这段时间内**发送的消息**。同样地，在机器的活动期间，从只读通信磁带上读出的内容称为（在这段时间内）**接收的消息**。

---

① 参见本章结束处进一步阅读的建议。

(不失一般性,在两个通信磁带上机器的活动都是单向的,比如说,从左到右。)

这个定义看起来不是很直观。特别地,有人可能会说一旦机器变为空闲的,就永远不会再变为活动的。也可能有人想知道将只读通信磁带和输入磁带进行区分(相应地,将只写通信磁带和输出磁带进行区分)的理由。答案是我们永远不会只考虑单个的交互机,而是考虑一对组合在一起并且一部分磁带相互重合的机器。直观地说,从一个交互机发出的消息将会由另一个与它共享通信磁带的交互机接收(因此,一个交互机的只读通信磁带就与另一个交互机的只写磁带重合)。通过改变共享交换磁带的内容可以使活动交互机变为空闲的,从而使另一台交互机(具有相反标识)变为活动的。这样一对交互机的计算包括,选定的交互机在初始(公共的)输入、(不同的)随机输入以及目前每个交互机接收到的消息的基础上进行的消息传递。

定义 4.2.2(两个 ITM 之间的连接计算):

- 两个交互机称为是**链接的**,如果它们拥有相反的标识、相同的输入磁带、相同的交换磁带,并且一台交互机的只读通信磁带是另一台交互机的只写通信磁带,或相反。我们要指出的是这两个交互机的其他磁带(即随机磁带、工作磁带和输出磁带)是不同的。
- 一对链接 ITM 在公共输入 x 上的**连接计算**是指一个表示两台机器本地配置的对序列。也就是说,每一对都包含两个字符串,每个字符串表示两台机器中某一台的本地配置。在每个这样的本地配置对中,一台机器(不一定总是特定的一台)是活动的,而另一台是空闲的。这个序列中的第一对包括对应于公共输入 x 的初始配置,并将交换磁带的内容设置为 0。
- 如果一台交互机停止了而交换磁带仍然保持着它的标识,我们说这两台交互机都停止了。就在这时确定两个机器的输出。

对于这一点,读者可能会反对这个定义,可能会说每个交互机都被剥夺了各自的本地输入(给予交互机的是各自不共享的随机磁带)。在 4.2.4 小节中将去掉这个限制,实际上允许各自的本地输入(除了公共的共享输入之外)是相当重要的(至少从实践的目的来看是这样)。然而,为了说明交互证明概念的作用,在第一次引进交互证明时,我们会倾向于选择前面较为简单的定义。另一方面,各自随机磁带的约定对于交互证明的作用是很重要的(见习题 4)。

4.2.1.2 与交互机相关的约定

典型地,我们考虑在每个交互机随机磁带的内容都(在所有的无限比特序列中)均匀独立选择时的执行情况。具有内部投币机制的无穷序列的约定不会让读者感到困扰,因为在一个有限的计算过程中,只会读取这个序列前面有限长的(起作用的)一段。每个随机磁带的内容都可以被看作是相应机器的内部硬币投掷(就像在第 1 章中通用概率机的定义一样)。因此,交互机实际上是概率的。

注释 令 A 和 B 是一对链接的 ITM,并假设 A 和 B 对每个公共输入的所有可能交互都会在有限步之后终止。我们用 $\langle A,B\rangle(x)$ 表示当均匀独立选择随机输入时,对公共输入 x,B 与交互机 A 交互(局部)输出的随机变量。(实际上这个定义是不对称的,因为它仅仅考虑了 B 的输出。)

另一个重要的约定是将交互机的时间复杂性只当作它的输入长度的函数。

定义 4.2.3(交互机的复杂性):我们说交互机 A 具有时间复杂度 $t:\mathbb{N}\to\mathbb{N}$,如果对于每一个交互机 B 和每一个字符串 x,都有当 A 与 B 关于公共输入 x 进行交互时,机器 A 总能(即,

无论 A 的随机磁带的内容和 B 的随机磁带的内容是什么)在 $t(|x|)$ 步内结束。特别地，如果存在一个多项式 p 可以使 A 具有时间复杂度 p，我们就称 A 具有多项式时间复杂度。特别地，我们说 A 是多项式时间的，如果存在一个多项式 p 满足 A 具有时间复杂度 p。

我们要强调的是，这样定义的时间复杂度与机器 A 接受消息的内容是独立的。换句话说，它是所有可能输入消息(也包括所有的内部硬币投掷)的一个上限。特别地，一个具有时间复杂度 $t(\cdot)$ 的交互机器，在输入 x 的情况下，可以只阅读发送给它的消息前面一段长度为 $t(|x|)$ 的片断。

4.2.1.3　证明系统

通常，证明系统用验证过程(可以视为叫做验证者的实体)的形式进行定义。针对特定论断的"证明"经常被认为来自外界(可以视为叫做证明者的另一个实体)。验证程序本身并不产生"证明"，而只是验证证明的合法性。交互证明系统的目的是通过与外界的交互能够得到被有效证明的结果。通常情况下，与外界之间的交互可能会很复杂，也可能会包括许多消息交互，只要验证者花费的总时间是(以公共输入为变量的)多项式时间。

鉴于采用了具有概率多项式时间算法的有效验证过程，考虑概率多项式时间验证者的选择是合理的。此外，验证者决定是否接受论断是随机的，并且允许有一个有限的错误概率(当然这些错误在重复验证过程足够多次后会减小到可以忽略的程度)。

不严格地说，我们需要证明者能够使验证者相信有效论断的有效性，而任何人都不能欺骗验证者，使他相信假的论断。这两个条件的概率解释是：需要验证者以"高"概率接受有效论断，而接受假论断的概率却"很小"(与验证者交互的交互机无关)。在下面的定义中，验证者的输出是指它接受或者拒绝公共输入的决定。输出 1 就代表"接受"，而输出 0 代表"拒绝"。

定义 4.2.4(交互证明系统)：一对交互机 (P,V) 叫做**语言 L 的交互证明系统**，如果交互机 V 是多项式时间并满足如下两个条件：

- 完备性：对于每个 $x\in L$，

$$\Pr[\langle P,V\rangle(x)=1]\geq\frac{2}{3}$$

- 合理性：对于每个 $x\notin L$ 和每个交互机 B，

$$\Pr[\langle B,V\rangle(x)=1]\leq\frac{1}{3}$$

以下是一些注释。我们首先要强调的是合理性条件涉及所有可能的"证明者"，而完备性条件仅仅涉及证明者 P。其次，验证者要求是(概率)多项式时间的机器，而证明者的计算能力没有资源方面的限制(在完备性条件和合理性条件中一样都不受限制)。第三，与 ***BPP*** 情况下类似，可以通过多次(多项式次)重复交互过程的方式来让上面定义的错误概率指数级减小。

在 ***NP*** 中的每种语言都有交互证明系统。特别地，令 $L\in \boldsymbol{NP}$，R_L 是关于语言 L 的证明关系(即，R_L 是多项式时间可分辨的，L 等同于集合 $\{x:\exists y$ 满足 $|y|=\text{poly}(x)\wedge(x,y)\in R_L\}$)。那么，语言 L 的交互证明包括一个对于输入 $x\in L$ 发送论断 y(同前)的证明者，和一个收到 y(对于公共输入 x)并且当 $|y|=\text{poly}(|x|)$，$(x,y)\in R_L$ 时输出 1(否则就输出 0)的验证者。很显然，在与指定的证明者进行交互的时候，验证者总是接受指定语言的输入。另一方面，不管"证明者"怎样欺骗，验证者永远都不会接受不属于指定语言的输入。我们要指出

的是，在这个证明系统中，证明者和验证者都是确定的（也就是说，他们都没有使用随机磁带）。显而易见，只有在 ***NP*** 中的语言才具有证明者和验证者都确定的交互证明系统（见习题 2）。

换句话说，***NP*** 可以被看作是一类交互证明系统，在这类系统中交互是单向的（也就是说，只是从证明者到验证者），验证者是确定的（并且不会出错）。在一般的交互证明中，没有这两个限制：交互是双向的，验证者是不确定的（可能会出现小概率的错误）。双向交互和随机性对于交互证明系统的作用都是重要的（参见习题 2）。

定义 4.2.5（*IP* 类）：由所有具有交互证明系统的语言构成的类称为 ***IP*** 类。

由前面的讨论可知 ***NP*** $\subseteq$ ***IP***。因为 ***BPP*** 中的语言都可以被视为具有不需要任何交互就可以确定成员的验证者，因此 ***BPP*** $\cup$ ***NP*** $\subseteq$ ***IP***。我们要提醒读者注意的是，目前还不知道 ***BPP*** $\subseteq$ ***NP*** 是否成立。

我们下面说明如果我们用两个函数 $\mathrm{c},\mathrm{s}: \mathbb{N} \to [0,1]$，满足 $\mathrm{c}(n) < 1-2^{\text{poly}(n)}$，$\mathrm{s}(n) > 2^{\text{poly}(n)}$，$\mathrm{c}(n) > \mathrm{s}(n) + \frac{1}{\text{poly}(n)}$ 来代替完备性条件和合理性条件确定的（常数）界，那么 ***IP*** 类的定义保持不变。也就是说，下面我们要考虑定义 4.2.4 的推广情况。

定义 4.2.6（广义交互证明）：令 $\mathrm{c},\mathrm{s}: \mathbb{N} \to \mathbb{R}$ 是对于某个多项式 $p(\cdot)$ 满足 $\mathrm{c}(n) > \mathrm{s}(n) + \frac{1}{p(n)}$ 的函数。交互对 (P,V) 叫做语言 L 的满足**完备性界** $\mathrm{c}(\cdot)$ 和**合理性界** $\mathrm{s}(\cdot)$ 的（广义）交互证明系统，如果

- （修正）完备性：对于每个 $x \in L$，
$$\Pr[\langle P,V\rangle(x) = 1] \geq \mathrm{c}(|x|)$$
- （修正）合理性：对于每个 $x \notin L$ 和每个交互机 B，
$$\Pr[\langle B,V\rangle(x) = 1] \leq \mathrm{s}(|x|)$$

定义为 $g(n) \stackrel{\text{def}}{=} \mathrm{c}(n) - \mathrm{s}(n)$ 的函数 $g(\cdot)$ 称为 (P,V) 的可接受间距，定义为 $\mathrm{e}(n) \stackrel{\text{def}}{=} \max\{1-\mathrm{c}(n), \mathrm{s}(n)\}$ 的函数 $\mathrm{e}(\cdot)$ 称为 (P,V) 的错误概率。特别地，s 是 (P,V) 的合理性错误，而 $1-\mathrm{c}$ 是它的完备性错误。

我们强调，c 是一个下界，而 s 是一个上界。

命题 4.2.7：下面 3 个条件等价：

1. $L \in$ ***IP***。也就是说，对于语言 L，存在一个具有完备性界 2/3 和合理性界 1/3 的交互证明系统。

2. L 具有强交互证明系统：对于每个多项式 $p(\cdot)$，语言 L 都存在一个以 2 为错误概率上界的交互证明系统。

3. L 具有弱交互证明：对于语言 L，存在一个多项式 $p(\cdot)$ 和一个以 $1/p(\cdot)$ 为可接受间距下界的广义交互证明系统。而且，这个系统的完备性界和合理性界（即数值 $\mathrm{c}(n)$ 和 $\mathrm{s}(n)$）可以在 n 的多项式时间内进行计算。

显然，前两个条件中的任何一个都可以推导出第 3 个条件（包括有效计算界限需要的条件）。有效计算完备性和合理性界限的能力也能用来证明相反（非退化）方向。证明留作练习（习题 1）。

4.2.2 一个实例（*IP* 中的图非同构问题）

到目前为止，我们所举的所有交互证明系统的例子都是退化的（例如，交互基本上都是单

向的)。现在我们举一个非退化交互证明系统。而且,我们所举的例子是不一定在 $\mathcal{BPP}\cup\mathcal{NP}$ 中的语言的交互证明系统。特别地,这种语言是非同构图对构成的集合 GNI。这个证明系统的想法源自下面的故事。

Petra von Kant 说大瓶的 Goldstar[①] 啤酒和小瓶的 Goldstar 啤酒喝起来不一样。Virgil 不相信她所说的话。为了证明她说的是对的,Petra 和 Virgil 多次重复如下过程来使得 Virgil 相信这个事实。

Virgil 随机选择一个大瓶的或者小瓶的啤酒,并将啤酒倒进用来尝酒的杯子中,但是不让 Petra 看到他用的是哪个瓶子。然后 Virgil 就将杯子递给 Petra,让她说出他刚才用的是哪个瓶子。

如果 Petra 的回答没有任何错误,那么 Virgil 就会相信她所说的一切。(但事实上,他应该在她正确回答了 50%以上时就相信她所说的话,因为如果不同瓶子中的啤酒喝起来是一样的,那么 Petra 应该没有办法以 50%以上的概率猜对所用的瓶子。)

现在我们回到正题上来。首先定义我们关心的语言:两个图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$ 称为是同构的,如果存在一个从顶点集 V_1 到顶点集 V_2 的 1－1 的映上的映射 π 满足:$(u,v)\in E_1$,当且仅当$(\pi(v),\pi(u))\in E_2$。如果映射 π 存在,就称其为这两个图之间的同构。非同构图对构成的集合记作 GNI。

构造 4.2.8(非同构图的交互证明系统):

- 公共输入:一对图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$。不失一般性,假设 $V=\{1,2,\ldots,|V_1|\}$,对 $V2$ 也可以进行相同的假设。
- 验证者的第一步(V1):验证者随机地在两个输入图中选择一个,并将这个图的一个随机同构拷贝发送给证明者。也就是说,验证者均匀地选择 $\sigma\in\{1,2\}$,并从顶点集合 V_σ 的置换集中随机地选择一个置换 π。验证者用顶点集 V_σ 和边集

$$F\stackrel{\text{def}}{=}\{(\pi(u),\pi(v)):(u,v)\in E_\sigma\}$$

  构造一个图,并将(V_σ,F)发送给证明者。
- 动机说明:就像证明者声称的那样,如果输入的两个图是不同构的,证明者就应该能够将一个图的同构拷贝和另一个图的同构拷贝区别开来(不必用高效的算法)。然而,如果输入的两个图是同构的,那么一个图的随机同构拷贝与另一个图的随机同构拷贝将会具有同样的分布。
- 证明者第一步(P1):当从验证者那里收到一个图 $G=(V',E)$时,证明者会找到 $\tau\in\{1,2\}$,使得图 G 与输入的图 G_τ 是同构的(如果 $\tau=1$ 与 $r=2$ 都满足这个条件,那么就任意选取 τ;如果 $\tau\in\{1,2\}$ 都不满足这个条件,就将 τ 设置为 0)。证明者将 τ 发送给验证者。
- 验证者第二步(V2):如果从证明者那里接收到的消息 τ 与 σ(在步骤 $V1$ 中选定的)相等,那么验证者输出 1(也就是说,接受公共输入)。否则验证者输出 0(即拒绝公共输入)。

这个验证者程序在概率多项式时间中很容易实现。我们并不知道证明者程序的概率多项式时间的实现,但这不是必需的。现在我们来说明前面的交互机对构成了 GNI(图非同构)语

① Goldstar 是一种以色列啤酒,有 330ml 和 500ml 两种规格。实际上,这个故事可以追溯到雅典娜关于神酒罐的声明,这是宙斯出的难题。不幸的是,奥维德并没有将他们交互的结果公开。

言中的(通常意义上的)交互证明系统。

命题 4.2.9:语言GNI属于 ***IP*** 类。而且,构造4.2.8中提出的程序构成了GNI语言的一个具有完备性界1和合理性界$\frac{1}{2}$的广义交互证明系统。即

1. 如果G_1和G_2不同构(即$(G_1,G_2)\in$ GNI),那么验证者(与证明者交互时)总是会接受。

2. 如果G_1和G_2同构(即$(G_1,G_2)\notin$ GNI),那么不管验证者与哪个机器交互,它都会以至少$\frac{1}{2}$的概率拒绝输入。

证明:显然,如果G_1和G_2不同构,那么就没有同时与G_1和G_2都同构的图。从而存在一个惟一的τ,使得图G'(步骤P1中证明者收到的)与输入图G_τ同构。因此,证明者在步骤P1中找到的τ总是与步骤V1中选择的σ相等。这样就证明了第一步。

另一方面,如果G_1与G_2同构,那么图G'与输入的两个图都是同构的。而且,在这种情况下,图G'不产生任何有关σ的信息,因此,没有机器可以(在输入G_1,G_2和G'的前提下)设置τ,使得它与σ相等的概率大于1/2。下面是证明的细节。

令π是图$G=(V,E)$的顶点集上的一个置换。我们用$\pi(G)$表示具有顶点集V和边集$\{(\pi(u),\pi(v)):(u,v)\in E\}$的图。令$\xi$是一个在$\{1,2\}$上服从均匀分布的随机变量,令$\Pi$是在$V$的置换集上服从均匀分布的随机变量。我们强调,这两个随机变量是相互独立的。我们感兴趣的是随机变量$\Pi(G_\xi)$的分布。我们将会说明尽管$\Pi(G_\xi)$是由随机变量Π和ξ决定的,但是随机变量ξ和$\Pi(G_\xi)$却是统计独立的。事实上,有如下命题成立。

结论 4.2.9.1:如果图G_1和G_2是同构的,那么对于每个与G_1(和G_2)同构的图G',都有:

$$\Pr[\xi=1\mid\Pi(G_\xi)=G']=\Pr[\xi=2\mid\Pi(G_\xi)=G']=\frac{1}{2}$$

证明:我们首先证明集合$S_1\stackrel{\text{def}}{=}\{\pi:\pi(G_1)=G'\}$和$S_2\stackrel{\text{def}}{=}\{\pi:\pi(G_2)=G'\}$的基数是相等的。这通过在集合$S_1$和集合$S_2$之间存在一个1—1的映上的对应很容易得出(这个对应是由图G_1和G_2之间的同构确定的)。因此,

$$\begin{aligned}\Pr[\Pi(G_\xi)=G'\mid\xi=1]&=\Pr[\Pi(G_1)=G']\\&=\Pr[\Pi\in S_1]\\&=\Pr[\Pi\in S_2]\\&=\Pr[\Pi(G_\xi)=G'\mid\xi=2]\end{aligned}$$

运用贝叶斯法则,就得到了结论。□

直观来讲,结论4.2.9.1是说对于每一个同构图对(G_1,G_2),随机变量$\Pi(G_\xi)$不产生关于ξ的任何信息,因此没有证明者能够以大于$\frac{1}{2}$的概率欺骗验证者接受错误。特别地,我们令R是任意一个随机的过程(代表一个基于(G_1,G_2)进行欺骗的证明者的可能策略),它在步骤V1中给定验证者消息的情况下试图猜出ξ的实际值。那么$R(\Pi(G_\xi))=\xi$代表验证者接受的事件,并且我们有

$$\Pr[R(\Pi(G_\xi))=\xi]=\sum_{G'}\Pr[\Pi(G_\xi)=G']\cdot\Pr[R(G')=\xi\mid\Pi(G_\xi)=G']$$

对第3个等式运用命题4.2.9.1,(对$\Pi(G_\xi)$支持的任何G')我们有

$$\begin{aligned}\Pr[R(G')=\xi\mid\Pi(G_\xi)=G']&=\sum_{v}\Pr[R(G')=v\,\&\,\xi=v\mid\Pi(G_\xi)=G']\\&=\sum_{v}\Pr[R(G')=v]\cdot\Pr[\xi=v\mid\Pi(G_\xi)=G']\end{aligned}$$

$$= \sum_{v \in \{1,2\}} \Pr[R(G') = v] \cdot \Pr[\xi = v]$$

$$= \frac{\Pr[R(G') \in \{1,2\}]}{2}$$

$$\leq \frac{1}{2}$$

当 R 总输出集合$\{1,2\}$中的元素时，取等号。这样就证明了第 2 部分。

构造 4.2.8 的注释：在构造 4.2.8 的证明系统中，验证者总是接受 GNI 语言中的输入（也就是说，在这样的情形下，错误概率等于 0）。$GNI \in \mathcal{IP}$，但并不知道 $GNI \in$ ***NP*** 是否成立这一事实是定理证明过程中交互证明能力和随机性的一个暗示。最后，我们注意证明者不知道验证者内部的硬币投掷的结果是重要的。要知道关于这些问题的进一步描述，请参考下面一节。

4.2.3* *IP* 类的结构

作为前面注释的继续，我们将会简洁地讨论与交互证明系统中的“证明能力”有关的几个方面的问题：

1. 完备性界和合理性界：本书中所提到的所有交互证明系统都恰好具有非常好的完备性；也就是说，验证者总是接受语言中的输入（即，完备性错误为 0）。实际上，任何交互证明系统都可以转换为验证者总是接受语言中的输入的（同一语言的）交互证明系统。

另一方面，如习题 5 所示，只有 ***NP*** 中的语言具有验证者总是拒绝不在语言中的输入的交互证明系统（即，合理性错误等于 0）。

2. 验证者硬币的保密性：*Arthur-Merlin* 证明（即，包括公开投币证明系统）是交互证明的一个特例，在这个特例中验证者必须发送他所投掷硬币的所有结果（因此就不需要发送其他消息了）。如前所述，构造 4.2.8 中的证明系统不是公开投币类型的。但是可以将任何交互证明系统转换为一个（同一语言的）公开投币交互证明系统，同时保持良好的完备性。

3. 哪些语言具有交互证明系统呢？（到现在为止我们一直忽略了这个自然的问题。）已经证明在 ***PSPACE*** 中的每种语言都具有交互证明系统。事实上，

IP 等于 ***PSPACE***

我们相信 ***PSPACE*** 比 ***NP*** 大得多；特别地，co***NP*** $\subseteq$ ***PSPACE***，而一般认为 co***NP*** $\neq$ ***NP***。而且，因为 ***PSPACE*** 在求补运算下是封闭的，所以 ***IP*** 也是这样。

4. 常数轮交互证明：构造 4.2.8 构成了一个常数轮协议（即，发送消息的数量是固定的）。相反地，在一般的 ***PSPACE*** 中的交互证明系统中，通信轮数是与输入的长度多项式相关的。我们相信 co***NP*** 不包含常数轮交互证明。

我们指出任何具有常数轮交互证明系统的语言也具有公开投币交互证明系统，而且这种公开投币交互证明系统只发送两个消息：其中后发送的消息包括验证者要求证明者回答的随机挑战。一般地，对于任何函数 $r: \mathbb{N} \to \mathbb{N}$，任何 $2r$ 轮证明系统都能够转换为一个（同一语言的）r 轮证明系统。

4.2.4 模型的扩展

为了使 4.3 节和 4.4 节中的描述更清晰，我们通过允许交互的双方都有各自的私有输入（作为公共输入的补充）来扩展交互证明系统的基本定义。不严格地说，这些输入是用来刻划交互的每一方的额外信息的。特别地，当用交互证明系统作为大协议中的子协议的时候，私有输入就与进入子协议前机器的本地配置有关。特别地，证明者的私有输入可能包含使证明者的任务有效完成的信息。

定义 4.2.10（修订的交互证明系统）：

1. 交互机的定义与定义 4.2.1 中的定义相同，并且该交互机具有一个额外的叫做**辅助输入磁带**的只读磁带。辅助输入磁带的内容叫做**辅助输入**。

2. 这样定义的交互机的复杂性仍然用（公共）输入长度的函数来衡量。也就是说，交互机 A 具有时间复杂性 $t:\mathbb{N}\to\mathbb{N}$，如果对于每个交互机 B 和每个字符串 x，交互机 A 与交互机 B 的交互过程总能（无论交互机 A 和交互机 B 的随机磁带和辅助输入磁带的内容是什么）在 $t(|x|)$ 步内结束。

3. 我们用 $\langle A(y),B(z)\rangle(x)$ 来表示对于公共输入 x，每个机器的随机输入均匀独立选择，A 具有辅助输入 y，B 具有辅助输入 z，B 与 A 交互时（本地）输出的随机变量。

4. 交互机对 (P,V) 叫做语言 L 的交互证明系统，如果机器 V 是多项式时间的并且满足下面两个条件：

- 完备性：对于每个 $x\in L$，都存在一个字符串 y，使得对于每个 $z\in\{0,1\}^*$，都有

$$\Pr[\langle P(y),V(z)\rangle(x)=1]\geq\frac{2}{3}$$

- 合理性：对于每个 $x\notin L$、每个交互机器 B 及每个 $y,z\in\{0,1\}^*$，都有

$$\Pr[\langle B(y),V(z)\rangle(x)=1]\leq\frac{1}{3}$$

要强调的是，当说一个交互机是多项式时间的时候，我们指的是它的运行时间是公共输入长度的多项式。因此，不能保证这样的机器具有足够的时间来读取它的所有辅助输入。

教学提示 交互证明的扩展模型在本书中第一次用到是在 4.3.3 小节中，此处零知识的概念扩展为验证者可以拥有先验信息。因此有人喜欢在给出零知识的基本定义之后给出定义 4.2.10，也就是把定义 4.2.10 延迟到 4.3.3 小节给出（然而，从概念上说，定义 4.2.10 确实是属于本节的）。

4.3 零知识证明：定义

在这一节中，我们介绍零知识交互证明系统的概念，并且举一个非退化的例子（具体地说，是对“下面的两个图是同构的”形式的声明）。

4.3.1 完备零知识和计算零知识

不严格地讲，我们说语言 L 的交互证明系统 (P,V) 是零知识的，如果对于输入 $x\in L$ 与 P 交互后有效计算的结果也能够从 x 有效计算出来（没有任何交互）。我们强调，上面的结论对于任何与 P 交互的有效方式都成立，而不仅仅是验证程序 V 中定义的方式。事实上，零知识是指定的证明者 P 具有的一个属性。它刻划的是证明者防止他人利用交互获得知识的健壮性。我们这种不正式的讨论可以用下面更为直接的方式进行描述。

令 (P,V) 是某种语言 L 的交互证明系统。我们说 (P,V)，事实上是 P，是完备零知识的，如果对于每个概率多项式时间交互机 V^* 都存在一个（普通）概率多项式时间算法 M^*，满足对于每个 $x\in L$，下面两个随机变量同分布：

- $\langle P,V^*\rangle(x)$（即，对于公共输入 x，交互机 V^* 在与交互机 P 交互后的输出）
- $M^*(x)$（即，对于输入 x，机器 M^* 的输出）

机器 M^* 叫做 V^* 与 P 交互的模拟器。

我们强调，我们要求对于每个和 P 交互的 V^*，而不仅仅是 V，都存在一个（"完备的"）模拟器 M^*。这个模拟器尽管不与交互机 P 交互，但是能模拟 V^* 与 P 的交互。这样的模拟器存在的事实意味着 V^* 从 P 那里没有获得任何知识（因为同样的输出可以不通过与 P 交互来产生）。

模拟规范

前面的讨论遵循的是在本书（特别是在第二卷中）其他章节中也要使用的通用定义规范。模拟规范要求交互的一方自己单独能做的任何事情都不能是与外界交互后的获益。这个规范的有效性是显然的，因为我们知道当我们说"做"的时候，指的是"有效地做"某些事情（同样地，"单独做某件事情"的复杂性与"在与外界交互后做某件事情"是紧密相关的）。[①] 诚然，与外部交互模拟的失败并不一定意味着交互以某些"真实获益"（从直觉的意义上）作为结果。因此问题的关键是当我们能够成功模拟时不能产生任何"真实获益"。总之，模拟规范所用的方法可能是过度慎重的，但是一定要是有效的（而且，至少应该说，提出一个"真实获益"的确切定义要难得多）。

退化情形 注意到 **BPP** 中的每种语言都有一个完备零知识证明系统，在这样的零知识证明系统中证明者什么都不做（验证者自己决定是接受还是拒绝公共输入）。为了描述零知识的这种"哑证明者"特性，我们可以给每个验证者 V^* 找到一个与它本质上相同的模拟器 M^*（区别在于，V^* 从不使用的通信磁带被看作是 M^* 的普通工作磁带）。

4.3.1.1 完备零知识

不幸的是，前面（完备）零知识的公式化描述有点太严格了（至少就我们所知）。[②] 我们通

---

① 参见 4.4.4.2 节中知识紧密性的讨论。

② 也就是，我们不知道满足这些条件的非退化的例子。相反，满足下面将给出的弱化定义的非退化例子我们是知道的，并且我们实际上也给出了一个（即，图同构的完备零知识证明）。

过允许模拟器在交互过程中以一定界限限定的概率失败来弱化上面的公式化描述。

定义 4.3.1(完备零知识)：令(P,V)是某种语言 L 的交互证明系统。我们说(P,V)是**完备零知识的**，如果对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，使得对于每个 $x\in L$，如下两个条件成立：

1. 对于输入 x，M^* 输出一个特殊符号(记为$\perp$)的可能性至多为$\frac{1}{2}$(即 $\Pr[M^*(x)=\perp]\leq\frac{1}{2}$)。

2. 令 $m^*(x)$为描述 $M^*(x)$在 $M^*(x)\neq\perp$的条件下分布的随机变量(即，对于每个 $\alpha\in\{0,1\}^*$，$\Pr[m^*(x)=\alpha]=\Pr[M^*(x)=\alpha|M^*(x)\neq\perp]$)。那么下面的随机变量同分布：

- $\langle P,V^*\rangle(x)$(即，对于公共输入 x，在与交互机 P 交互之后交互机 V^* 的输出)
- $m^*(x)$(即，对于输入 x，在 M^* 不是$\perp$的条件下 M^* 的输出)

机器 M^* 叫做 V^* 与 P 交互的**完备模拟器**。

条件 1 可以被另一个更强的条件取代，即要求 M^* 以可以忽略的概率输出特殊符号。例如，可以要求(对于输入 x)交互机 M^* 输出$\perp$的概率的上界为 $2^{-p(|x|)}$，其中 $p(\cdot)$为任何多项式；参考习题习 6。因此，随机变量$\langle P,V^*\rangle(x)$和 $M^*(x)$的统计差别可以变得忽略不计(关于$|x|$)；参考习题 8。因此，无论验证者在同证明者交互以后计算出什么结果，这些结果(验证者自己)都可以通过模拟器计算出来(二者的结果之间仅有极小的误差)。

4.3.1.2 计算零知识

遵循第 3 章的精神，我们注意到从实践的角度看，没有必要"完全模拟"与 P 交互之后 V^* 的输出。事实上，产生一个与 P 交互后 V^* 的输出计算不可分辨的概率分布就足够了。这样的弱化与我们初始的要求"对于输入 $x\in L$，与 P 交互后有效计算的任何结果都可以从 x 有效计算出来(没有任何交互)"是一致的，原因是我们把计算不可分辨的总体看作是相同的。在提出一般零知识的弱化定义之前，我们回想一下计算不可分辨总体的定义(参见定义 3.2.2 中的条目 2)。这里我们考虑以语言 L 中的字串为下标的总体。我们说总体$\{R_x\}_{x\in L}$和$\{S_x\}_{x\in L}$是**计算不可分辨的**，如果对于每个概率多项式时间算法 D、每个多项式 $p(\cdot)$和所有足够长的 $x\in L$，都有：

$$|\Pr[D(x,R_x)=1]-\Pr[D(x,S_x)=1]|<\frac{1}{p(|x|)}$$

定义 4.3.2(计算零知识)：令(P,V)是某种语言 L 的一个交互证明系统。我们说(P,V)是**计算零知识的**(或者就说是**零知识的**)，如果对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，使得下面两个总体是计算不可分辨的：

- $\{\langle P,V^*\rangle(x)\}_{x\in L}$(即，对于输入 x，交互机 V^* 与交互机 P 交互后的输出)
- $\{M^*(x)\}_{x\in L}$(即，对于输入 x，机器 M^* 的输出)

机器 M^* 叫做 V^* 与 P 交互的**模拟器**。

读者可以很容易地验证(参看习题 9)，允许模拟器(以不大于$\frac{1}{2}$的概率)输出特殊符号，而且考虑条件输出分布(如定义 4.3.1)并不增加定义 4.3.2 的能力。

零知识的范围 我们强调，零知识的两个定义都适用于通常意义上(即，在语言内输入与语言外输入的接受概率之间存在不可忽略的间距)的交互证明系统。事实上，零知识的定义可应用于任何一对交互机(实际上可应用于每个交互机)；也就是说，我们可以说交互机 A 在 L

上是零知识的，如果对于公共输入 $x\in L$，与 A 交互后有效计算的任何结果都可以从 x 自身有效计算出来。

4.3.1.3 零知识的一个可选的公式化描述

零知识的一个可选的公式化描述考虑验证者对他与证明者交互的观点，而不仅仅考虑验证者与证明者交互后的输出。"验证者对交互的观点"指的是验证者和证明者交互(执行)过程中本地配置的完整序列。很清楚，只考虑验证者随机磁带的内容和验证者在执行过程中从证明者那里接收到的消息序列就足够了(因为本地配置的完整序列和最终输出是由这些对象决定的)。

定义 4.3.3(零知识，可选的公式化描述)：令(P,V)，L 和 V^* 与定义 4.3.2 中定义的相同。我们用 $\text{view}_{V^*}^{P}(x)$表示描述 V^* 的随机磁带内容和 V^* 在对公共输入 x 进行联合计算时从 P 接收到的消息的随机变量。我们说(P,V)是**零知识的**，如果对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，满足总体$\{\text{view}_{V^*}^{P}(x)\}_{x\in L}$和总体$\{M^*(x)\}_{x\in L}$是计算不可分辨的。

这里有一些注释。首先，注意到定义 4.3.3 是通过将定义 4.3.2 中的$\langle P,V^*\rangle(x)$替换为 $\text{view}_{V^*}^{P}(x)$得到的。定义 4.3.3 中使用的模拟器 M^* 与定义 4.3.2 中使用的模拟器是相关的，但不是相等的(尽管这个事实并没有在这两个定义中反应映出来)。显然，对于每一个 V^*，$\langle P,V^*\rangle(x)$都能用(确定的)多项式时间从 $\text{view}_{V^*}^{P}(x)$中计算出来。尽管相反的方向不一定总成立，定义 4.3.3 与定义 4.3.2 却是等价的(通过对 V^* 的通用量化；参考习题 10)。后面这个事实调整了定义 4.3.3 的使用，尽管定义 4.3.3 看起来不如定义 4.3.2 自然，但是却更方便使用。完备零知识的一个类似的可选公式化描述是显然的，并且它与定义 4.3.1 是等价的。

4.3.1.4* 几乎完备(统计)零知识

一个(比计算零知识)弱化程度更轻的完备零知识的弱化概念如下。

定义 4.3.4(几乎完备(统计)零知识)：令(P,V)是某种语言 L 的交互证明系统。我们说(P,V)是**几乎完备零知识的**(或者**统计零知识的**)，如果对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，使得如下两个总体作为$|x|$的函数在统计特性上接近：

- $\{\langle P,V^*\rangle(x)\}_{x\in L}$(即，公共输入为 x 时，交互机 V^* 在与交互机 P 交互后的输出)
- $\{M^*(x)\}_{x\in L}$(即，输入 x 时，机器 M^* 的输出)

也就是说，就$|x|$来说，$\langle P,V^*\rangle(x)$和 $M^*(x)$之间的统计差别是可以忽略不计的。

正如计算零知识中那样，允许模拟器(以不大于 1/2 的概率)输出特殊符号，并且考虑条件输出分布(如定义 4.3.1)并不增加定义 4.3.4 中证明的能力；参看习题 8。也很容易看出，完备零知识都是几乎完备零知识，而几乎完备零知识都是计算零知识。

这 3 个定义(即完备零知识、几乎完备零知识和计算零知识)与概率总体的"接近"对概念的 3 个层次的解释是对应的(在这 3 个定义中，要求接近的总体对都指的是$\{\langle P,V^*\rangle(x)\}_{x\in L}$和$\{M^*(x)\}_{x\in L}$)。

1. 接近的最严格解释是要求两个总体同分布。这与完备零知识情况下的要求一致。

2. 接近的严格性有所降低的解释是两个总体是统计不可分辨的(或者说是统计接近的)。这与几乎完备(或者统计)零知识情况下的要求一致。

3. 接近的能够满足所有实际要求严格性最低的解释是两个集合是计算不可分辨的。这与计算零知识情况下的要求一致。

4.3.1.5* 基于零知识的复杂度类

各种零知识的定义导致了自然的复杂度类。

定义 4.3.5(具有零知识证明的语言类):我们用 ***ZK***(也用 ***CZK***)表示具有(计算)零知识交互证明系统的所有语言构成的类。同样地,用 ***PZK***(***SZK***)表示具有完备(统计)零知识交互证明系统的所有语言构成的类。显然,

$$\boldsymbol{BPP} \subseteq \boldsymbol{PZK} \subseteq \boldsymbol{SZK} \subseteq \boldsymbol{CAK} \subseteq \boldsymbol{IP}$$

如果存在(非均匀)单向函数,则最后一个包含是一个等式(即 $\boldsymbol{CZK} = \boldsymbol{IP}$);参考命题 4.4.5 和定理 3.5.12 及定理 4.4.12。另一方面,我们相信第一个和第三个包含是严格的(因为等号成立与众所周知的复杂度假定矛盾)。因此,我们相信

$$\boldsymbol{BPP} \subset \boldsymbol{PZK} \subseteq \boldsymbol{SZK} \subset \boldsymbol{CAK} = \boldsymbol{IP}$$

PZK 与 ***SZK*** 之间的关系至今仍是未决问题(不管是⊂还是=,都没有证明)。

4.3.1.6* 期望多项式时间模拟器

定义 4.3.1 中提出的完备零知识的公式化描述与在一些早期公开文献中用到的定义是不同的。早期的定义要求模拟器总是输出一个合法的结果(该结果与实际交互的结果同分布),却允许模拟器以期望多项式时间而不是严格意义上的多项式时间来运行。我们强调,这种期望是对模拟器所有的投币而取的(模拟器的输入不变)。这就导致了下面的定义。

定义 4.3.6(自由形式的完备零知识):我们说(P,V)是**自由意义上完备零知识的**,如果对于每个概率多项式时间交互机 V^*,都存在一个期望多项式时间算法 M^*,满足:对于每个 $x \in L$,随机变量$\langle P,V^*\rangle(x)$和 $M^*(x)$都同分布。

我们强调,概率多项式时间指的是所有可能执行中运行时间的一个严格的界限,而期望多项式时间指的是允许有非多项式时间的执行,但是运行时间"在平均意义上是多项式的"。很明显,定义 4.3.1 暗示着定义 4.3.6(参考习题 7)。有趣的是,存在着与自由完备零知识定义的完备零知识一致的交互证明,却还未发现与定义 4.3.1 定义的完备零知识一致的交互证明。我们指出,将期望概率多项式时间算法变换为可以在严格的多项式时间内运行的算法的天真想法在当前的环境中是不适合的。[①]

我们倾向于采用定义 4.3.1 而不采用定义 4.3.6,因为我们想避开期望多项式时间的概念。我们这样做的理由是,在平均多项式时间与有效计算之间的对应关系要比已被广泛接受的严格多项式时间和有效计算之间的对应关系更有争议。此外,期望多项式时间的概念也更含混,不够一目了然。

期望多项式时间的直接解释是具有由输入长度的多项式所限定的平均运行时间。期望多

---

① 这种直接转换缩减了算法(这里我们指的是模拟器)的运行,而这个算法需要的步骤是预期的 t 倍还多(这种缩减运行可以产生固定的输出)。算法修改前后输出分布的统计差别至多为 $1/t$。问题是 t 必须由运行时间的固定多项式来限定,这样统计差别才不能忽略。为了说明我们对这种直接转换的分析是严密的,我们考虑下面算法转换的有效性:输入 1^n 时,算法首先平均选择 $r \in \{0,1\}^n$,然后执行 2^t 步空操作,最后运行 $S(1^n)$,这里 t 是 r 最长的全 0 前缀的长度,S 是任意(严格的)概率多项式时间算法。

项式时间的这个定义是不能令人满意的，因为它在约减运算(reductions)下不封闭并且太与机器相关了。这两个恶化现象源于这样的事实：一个函数可能会有以(n 的)多项式为界的均值(比如说，在$\{0,1\}^n$上的均值)，但是这个函数的平方函数却不以(n 的)多项式为界。例如，令函数 $f(x)\stackrel{\text{def}}{=}2^{|x|}$(当 $x\in\{0\}^*$ 时)，$f(x)\stackrel{\text{def}}{=}|x|^2$(当 $x\notin\{0\}^*$ 时)，则满足 $\mathsf{E}[f(U_n)]<n^2+1$，但$\mathsf{E}[f(U_n)^2]>2^n$。

因此，期望多项式时间的一个更好解释是具有以具有平均线性增长率的函数为变量构成的多项式为界的运行时间。也就是说，用平均线性的定义，我们说 f 是平均多项式的，如果存在一个多项式 p 和一个平均意义上的线性函数 l，使得对于所有足够长的 x，都有 $f(x)\leq p(l(x))$。注意，如果 f 是平均多项式的，那么 f^2 也是平均多项式的。

类似的讨论也适用于计算零知识。更特别地，定义 4.3.2 要求模拟器以多项式时间工作，而自由形式的概念允许模拟器以期望多项式时间工作。

为了工整，通常修改允许期望多项式模拟器的定义为要求期望多项式时间的验证者与证明者交互时也存在这样的模拟器。

4.3.1.7* 诚实验证者零知识

我们简要讨论一个零知识的较弱形式的概念。这个叫做诚实验证者零知识的概念要求只模拟指定的(或者诚实的)验证者的观点，而不是对任意可能的(概率多项式时间的)验证者的观点进行模拟。虽然这个概念对于典型密码学应用来说是不够的，但是它是有趣的，至少有下面两个理由：第一，这个零知识的弱概念是非常不平凡的和吸引人的；第二，具有诚实证明者的零知识公开投币协议可以转化为一般零知识的类似协议。我们强调，在当前的环境中(具有单个指定的验证者)，输出模拟(如定义 4.3.2 所述)和观点模拟(如定义 4.3.3 所述)的公式化描述并不相同，后者更为重要。[①]

定义 4.3.7(关于诚实验证者的零知识)：令(P,V)，L 和$\text{view}_V^P(x)$与定义 4.3.3 中定义的相同。我们说(P,V)是诚实验证者零知识，如果存在一个概率多项式时间算法 M，使得总体$\{\text{view}_V^P(x)\}_{x\in L}$和$\{M(x)\}_{x\in L}$是计算不可分辨的。

上面的定义与计算零知识有关，并且是对定义 4.3.3 的一个限制。完备零知识和统计零知识的相应版本可类似定义。

4.3.2 一个例子(*PZK* 中的图同构)

如前所述，***BPP*** 中的每种语言都有一个平凡的(即退化的)零知识证明系统。我们现在举一个非退化零知识证明系统的例子。而且，我们举的例子对应的语言不知道是否在 ***BPP*** 中。特别地，这种语言就是同构图对构成的集合，记为 GI(参考 4.2 节中的定义)。同样地，这个证明系统的思想可以通过下面的故事表达出来。

在这个故事中，Petra von Kant 声称在她的迷宫里有一条从北门到南门的小路(即穿越迷

---

① 注意，对于具有完全完备性的任何交互证明系统，诚实验证者的输出是很容易模拟的(通过一个总是输出 1 的算法来模拟)。相反，4.5 节中给出的许多负面结果也适用于下面定义的关于诚实验证者的零知识。例如，只有 ***BPP*** 中的语言具有关于诚实验证者零知识的单向证明系统。

宫内部的一条小路)，但是 Virgil 不相信她的话。Petra 想证明这件事给 Virgil 看，但是又不想告诉 Virgil 任何额外的知识(特别是不想帮他找到这条从南门到北门的小路)。为了证明这件事，Petra 和 Virgil 重复如下过程足够多次来使 Virgil 逐渐相信 Petra。

Petra 非常神奇地将 Virgil 送到迷宫中一个随机的地方，然后 Virgil 要求 Petra 告诉他去南门或者去北门的路。假定他的选择是随机的，但他可能试图去欺骗。Petra 随机地选择一条(足够长的)从当前位置到指定地点的路线，并带领 Virgil 沿着这条路线走出迷宫。

很明显，如果迷宫确实有 Petra 所说的小路的话(Petra 知道这条路)，那么 Virgil 就会相信 Petra 说的是真的。另一方面，如果迷宫没有这条路，在每次实验中至少有 50%的概率，Virgil 会发现 Petra 在撒谎。最后，Virgil 并没有从实验中获得知识，因为他能按照如下方式独自模拟这个试验：首先，他选择北门或南门(就像他在实验中那样)，(从迷宫外)走到这个门口。然后他选择一条随机的路线走到迷宫里，同时边走边释放一个线团。最后他沿着这条线回到门口。(长度等同于实验中由 Petra 所带领的那条路线长度的足够长的随机路线，这样能保证 Virgil 可以走到迷宫中的一个随机的地方，并且沿着这条路线反向走回来，看起来像是从他所在位置到指定门口的一条随机路线。)

现在我们回到前述的说明。

构造 4.3.8(同构图的完备零知识证明)：

- 公共输入：两个图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$。令 ϕ 为这两个输入图之间的同构映射；即，ϕ 是从顶点集 V_1 到顶点集 V_2 的 1－1 的映上的映射，并且满足：$(u,v)\in E_1$ 当且仅当 $(\phi(v),\phi(u))\in E_2$。
- 证明者的第一步(P1)：证明者随机选择 G_2 的一个同构拷贝，并且把它发送给验证者。即，证明者随机地从具有均匀概率分布的顶点集 V_2 的置换集中选择一个置换，构造一个具有顶点集 V_2 和边集 $F\stackrel{\text{def}}{=}\{(\pi(u),\pi(v)):(u,v)\in E_2\}$ 的图。证明者将 (V_2,F) 发送给验证者。
- 动机说明：如果如证明者声称的那样，输入的两个图是同构的，那么在步骤 $P1$ 中所发送的图与输入的两个图都同构。反之，如果输入的两个图不同构，则不存在与输入的两个图都同构的图。
- 验证者的第一步(V1)：当从证明者那里接收到一个图 $G'=(V',E')$ 后，验证者要求证明者出示在 G' 与验证者随机选择的一个输入图之间的同构。即验证者任意选择 $\sigma\in\{1,2\}$，将其发给证明者(证明者以 G_σ 和 G' 之间的同构来应答)。
- 证明者的第二步(P2)：如果从验证者那里接收到的消息 σ 等于 2，那么证明者将 π 发给验证者。否则(也就是当 $\sigma\neq 2$ 时)，证明者发送 $\pi\circ\phi$(即 π 的 ϕ 合成，定义为 $\pi\circ\phi(v)\stackrel{\text{def}}{=}\pi(\phi(v))$)给验证者(注意：证明者将任意 $\sigma\neq 2$ 都视为 $\sigma=1$)。
- 验证者的第二步(V2)：如果来自证明者的消息 ψ 是 G_σ 和 G' 之间的同构，则验证者输出 1；否则输出 0。

我们把证明者的程序记为 P_{GI}。

上面提到的验证者程序可以很容易地在概率多项式时间内实现。如果证明者有一个在输入的两个图之间的同构映射作为辅助输入，那么证明者程序也能在概率多项式时间内完成。我们现在来说明上面的交互机对构成了语言 GI(图同构)的一个(通常意义上的)零知识交互

证明系统。

命题 4.3.9：语言 GI 有完备零知识交互证明系统。而且，在构造 4.3.8 中详细说明的那些程序满足：

1. 如果 G_1 和 G_2 是同构的(即 $(G_1,G_2)\in GI$)，那么验证者(当和证明者交互时)总会接受。

2. 如果 G_1 和 G_2 不同构(即 $(G_1,G_2)\notin GI$)，那么无论验证者和哪个交互机交互，它拒绝接受输入的概率都不小于 $\frac{1}{2}$。

3. 证明者(即 P_{GI})是完备零知识的。即，对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*(M^* 输出 $\perp$ 的概率至多为 $\frac{1}{2}$)，使得对于每个 $x\stackrel{\text{def}}{=}(G_1,G_2)\in GI$，下面的两个随机变量都同分布：

- $\text{view}_{V^*}^{P_{GI}}(x)$(即对于公共输入 x，在与 P_{GI} 交互后 V^* 的观点)
- $m^*(x)$(即对于输入 x，在输出不为 $\perp$ 的条件下，M^* 的输出)

通过连续执行 k 次上面的协议，可以得到一个 GI 的具有差错概率 2^{-k}(仅在合理性条件下)的零知识交互证明系统。我们强调，在每次重复实现协议时，证明者和验证者都必须使用新的硬币投掷，当前的硬币投掷与前面的硬币投掷是独立的。更多的讨论请参考 4.3.4 小节。我们要说明的是，k 次重复执行上面的协议可以将合理性条件减少到 2^{-k}，但是，在 k 的增长速度快于输入长度的对数时，得到的交互证明不知道是不是零知识的。实际上，我们相信这样得到的交互证明系统不是零知识的。更多的讨论请参考 4.5 节。

我们强调，现在并不知道 $GI\in$ ***BPP*** 是否成立。因此，命题 4.3.9 给出的是存在对应于不知道是不是在 ***BPP*** 中的语言的完备零知识证明。

证明：我们首先说明，上面的程序实际上构成了一个 GI 的(通用)交互证明系统。很明显，如果输入的图 G_1 和 G_2 是同构的，那么由步骤 $P1$ 构造的图 G' 与这两个图都同构。因此，如果双方都按预定程序进行动作，那么验证者总会接受(即，输出 1)。这就证明了第一部分。另一方面，如果 G_1 和 G_2 不同构，则不存在与 G_1 和 G_2 都同构的图。因此无论证明者(可能是欺骗性的)如何构造图 G'，总是存在 $\sigma\in\{1,2\}$，使得 G' 和 G_σ 不同构。因此，当验证者按照预定的程序动作时，验证者拒绝(即，输出 0)的概率将不会少于 $\frac{1}{2}$。这样就证明了第二部分。

剩下的就是要证明，P_{GI} 实际上是 GI 上的完备零知识。这事实上是整个证明中最难的部分。模拟构造 4.3.8 中所示的验证者的输出是很容易的(因为对于语言 GI 中的输入，它的输出都是 1)。而且，除了在最后会输出和 P_{GI} 交互后的全部结果之外，模拟按照构造 4.3.8 中指定程序来做的验证者的输出也不难(参看习题 12)。难的是模拟随意背离指定程序的有效验证者的输出。

我们将在这里使用(完备)零知识的另一种公式化描述，并且说明对于每个概率多项式时间交互机 V^*，怎样来模拟 V^* 对 V^* 与 P_{GI} 交互的观点。如前所述，当验证者按指定程序动作的时候，模拟 V^* 和 P_{GI} 交互的观点并不难。然而，我们需要模拟在一般情况下验证者的观点(在这种情况下使用的是任意多项式时间交互程序)。下面是我们模拟(即对于每个 V^*，都构造一个模拟器 M^*)的概况。

模拟器 M^* 集成了交互程序 V^* 的代码。对于输入 (G_1,G_2)，模拟器首先随机选择一个输入图(G_1 或者 G_2)，然后产生这个图的一个随机同构拷贝，记为 G''。在这个过程中，模拟器与

P_{GI} 的行为不同，但是所产生的图（即 G''）与交互证明的步骤（P1）中所发送的消息是同分布的。比如说模拟器通过对 G_1 的随机置换产生了 G''。现在，如果 V^* 要求查看 G_1 和 G'' 之间的同构，则模拟器能够正确应答，并且在此过程中模拟器完成了验证者对与 P_{GI} 交互的观点的模拟。但是，如果 V^* 要求查看 G_2 和 G'' 之间的同构，则模拟器（与 P_{GI} 不同，模拟器并不"知道" ϕ）无法正确应答，我们让模拟器输出 ⊥ 并终止运行。我们强调，模拟器"无法得知"V^* 要验看与 G_1 的同构还是与 G_2 的同构。关键是模拟器能够随机地尝试两种可能中的一种，并且如果幸运的话（发生的概率为 $\frac{1}{2}$），它的输出与 V^* 在和 P_{GI} 交互时的观点（对于公共输入 (G_1,G_2)）同分布。一个关键的事实（参看稍后的命题 4.3.9.1）是 G'' 的分布与模拟器对输入图形的选择是随机独立的，因此不能影响模拟器幸运成功的概率。下面是模拟器的一种详细描述。

模拟器 M^*　对于输入 $x \stackrel{\text{def}}{=} (G_1,G_2)$，模拟器 M^* 按如下程序动作：

1. 设置 V^* 的随机磁带：令 $q(\cdot)$ 表示限定运行时间的多项式。模拟器 M^* 每次都通过均匀选择一个将被用作 V^* 随机磁带内容的字符串 $r \in \{0,1\}^{q(|x|)}$ 来启动（也可以在下面的步骤 3 中"空闲地"产生 V^* 的硬币）。

2. 模拟证明者的第一步（P1）：模拟器 M^* 随机选择一个具有均匀概率分布的"比特"$\tau \in \{1,2\}$ 和一个具有均匀分布的顶点集 V_τ 的置换 ψ。然后构造一个具有顶点集 V_τ 和边集 $F \stackrel{\text{def}}{=} \{(\psi(u),\psi(v)) : (u,v) \in E_\tau\}$ 的图，并且令 $G'' \stackrel{\text{def}}{=} (V_\tau, F)$。

3. 模拟验证者的第一步（V1）：模拟器 M^* 通过把 x 放到 V^* 的公共输入磁带里，把 r（在上面的步骤 1 中选择的）放到 V^* 的随机磁带里，把 G''（在步骤 2 中构造的）放到 V^* 的输入消息磁带里，来启动 V^* 的执行。在执行 V^* 的步骤多项式次后，模拟器能够阅读 V^* 的输出消息，记为 σ。为了简化下面的叙述，我们把 σ 标准化：$\sigma \neq 2$ 的值设置为 $\sigma = 1$，$\sigma = 2$ 的值保持不变。

4. 模拟证明者的第二步（P2）：如果 $\sigma = \tau$，则模拟器输出 (x, r, G'', ψ)，然后停止。

5. 模拟失败：如果 $\sigma \neq \tau$，模拟器输出 ⊥，然后停止。

因为 V^* 是多项式时间的，所以模拟器 M^* 也是多项式时间的。剩下要说明的是，M^* 输出 ⊥ 的概率至多为 $\frac{1}{2}$，在不输出 ⊥ 时，模拟器的输出与验证者对"与 P_{GI} 真正的交互"的观点同分布。上面的命题是下面这两个结论证明的关键。

结论 4.3.9.1：假定图 G_1 和 G_2 是同构的。令 ξ 为一个在 $\{1,2\}$ 上服从平均分布的随机变量，Π 为在 V_ξ 的置换集上服从均匀分布的随机变量。那么，对于每个与 G_1（和 G_2）同构的图形 G''，都有：

$$\Pr[\xi = 1 \mid \Pi(G_\xi) = G''] = \Pr[\xi = 2 \mid \Pi(G_\xi) = G''] = \frac{1}{2}$$

其中，与结论 4.2.9.1 中一样，$\pi(G)$ 表示使用置换 π 对 G 的顶点重新标号后得到的图。

结论 4.3.9.1 与结论 4.2.9.1（用来说明构造 4.2.8 构成了一个关于 GNI 的交互证明系统的命题）[①]是雷同的。正如命题 4.2.9 证明的剩余部分那样，给定 $\Pi(G_\xi)$，任意输出值域为 $\{1,2\}$ 的随机化过程输出 ξ 的概率都是 $\frac{1}{2}$。因此，给定 G''（模拟器在步骤 2 中构造的），验证者程序输出（标准化的）σ、$\sigma \neq \tau$ 的概率为 $\frac{1}{2}$。我们可以推断出模拟器输出的概率为 $\frac{1}{2}$。剩下要证

① 在构造 4.2.8 中，图 $\Pi(G_\xi)$ 被提交给证明者，结论 4.2.9.1 用于建立证明系统的合理性（即分析当 $(G_1,G_2) \notin GNI$ 时也就是 $(G_1,G_2) \in GI$ 时发生的事情）。而在此处 $\Pi(G)_\xi$ 被提交给验证者，命题被用于建立零知识特性（因此也讨论 $(G_1,G_2) \in GI$ 的情况）。

明的是，在模拟器不输出⊥时，模拟器的输出与“V^* 对真正交互的观点”相同。

结论 4.3.9.2：令 $x=(G_1,G_2)\in GI$。那么对于每个字符串 r、图 H 和置换 ψ，都有

$$\Pr\left[\text{view}_{V^*}^{P_{GI}}(x)=(x,r,H,\psi)\right]=\Pr[M^*(x)=(x,r,H,\psi)\mid M^*(x)\neq\perp]$$

证明：令 $m^*(x)$ 用来描述输出不为⊥的 M^*。我们首先注意到，$m^*(x)$ 和 $\text{view}_{V^*}^{P_{GI}}(x)$ 的分布都为四元组 $(x,r,\cdot,\cdot)$ 的形式，且 $r\in\{0,1\}^{q(|x|)}$ 服从均匀分布。令 $V(x,r)$ 为描述 $\text{view}_{V^*}^{P_{GI}}(x)$ 最后两个元素在第二个元素等于 r 的条件下的随机变量。同样地，令 $\mu(x,r)$ 为描述 $m^*(x)$ 最后两个元素的随机变量(在第二个元素等于 r 的条件下)。我们需要证明对于每个 x 和 r，$V(x,r)$ 和 $\mu(x,r)$ 是同分布的。注意，一旦 r 确定了，在公共输入为 x、随机磁带为 r、输入消息为 H 时，发送的消息也就惟一确定了。我们把这个消息记为 $v^*(x,r,H)$。

我们要证明，在集合

$$C_{x,r}\stackrel{\text{def}}{=}\{(H,\psi):H=\psi(G_{v^*(x,r,H)})\}$$

上，$V(x,r)$ 和 $\mu(x,r)$ 是同分布的，其中 $\psi(G)$ 表示将 G 的顶点通过随机置换 ψ 重新标号得到的图(即，如果 $G=(V,E)$，那么 $\psi(G)=(V,F)$，使得 $(u,v)\in E$ 的充要条件是 $(\psi(u),\psi(v))\in F$)。这个结论的证明冗长乏味且与本书内容无关(因此将其略去并无影响)。

上述证明很重要，因为它与图 G_2 的自同构群(即置换 π 的集合，满足 $\pi(G_2)$ 不但与 G_2 同构而且与 G_2 相同)有关。为了叙述简单，我们先来考虑一种特殊情况：G_2 的自同构群仅由恒等置换构成(即 $G_2=\pi(G_2)$ 当且仅当 π 是恒等置换)。在这种情况下，$(H,\psi)\in C_{x,r}$ 当且仅当 H 与(G_1 和)G_2 同构，且 ψ 是 H 与 $G_{v^*(x,r,H)}$ 之间(惟一的)同构映射。因此，$C_{x,r}$ 的元素个数为 $|V_2|!$，且每个元素中的第一个分量即图 H 各不相同。在一般情况下，$(H,\psi)\in C_{x,r}$ 当且仅当 H 与(G_1 和)G_2 同构，且 ψ 是 H 与 $G_{V^*(x,r,H)}$ 之间的同构映射。我们强调，对于所有包含 H 的对，$v^*(x,r,H)$ 都是相同的。令 $\text{aut}(G_2)$ 表示 G_2 的自同构群的元素个数。那么每个 H(与 G_2 同构)在 $C_{x,r}$ 中的 $\text{aut}(G_2)$ 个对中出现，且每个这样的对中，H 和 $G_{V^*(x,r,H)}$ 之间的同构映射均不同。与 G_2 同构且互不相同的 H 的个数为 $|V_2|!/\text{aut}(G_2)$，因此在一般情况下，$|C_{x,r}|=|V_2|!$。

我们首先考虑随机变量 $\mu(x,r)$(表示 m^* 的后缀)。回想一下 $\mu(x,r)$ 是由如下的两步随机处理过程确定的。第一步，随机地选择在集合($\{1,2\}\times$置换)上服从均匀分布的对(τ,ψ)，并且令 $H=\psi(G_\tau)$。第二步，如果 $v^*(x,r,H)=\tau$，则令 $\mu(x,r)=(\psi(G_\tau),\psi)$(否则忽略$(\tau,\psi)$对)。因此，在第一步中，通过 $\text{aut}(G_2)$ 个不同的$(1,\cdot)$对(即满足 $H=\psi(G_1)$ 的$(1,\psi)$对)，以及 $\text{aut}(G_2)$ 个不同的$(2,\psi)$对(即满足 $H=\psi(G_2)$ 的$(2,\psi)$对)，就产生了一个(与 G_2 同构的)图 H。所有这些 $2\times\text{aut}(G_2)$ 个对产生了同一个图 H，因而导致了同一个 $v^*(x,r,H)$ 值。因此，在产生图 $H=\psi(G_\tau)$ 的 $2\cdot\text{aut}(G_2)$ 个(τ,ψ)对中，只有 $\text{aut}(G_2)$ 个满足 $\tau=v^*(x,r,H)$ 的对能够产生输出。于是对于每个(与 G_2 同构的)H，$\mu(x,r)=(H,\cdot)$ 的概率等于 $\text{aut}(G_2)/(|V_2|!)$。而且，对于每个(与 G_2 同构的)H，都有

$$\Pr[\mu(x,r)=(H,\psi)]=\begin{cases}\dfrac{1}{|V_2|!} & \text{如果 } H=\psi(G_{v^*(x,r,H)})\\ 0 & \text{其它}\end{cases}$$

因此 $\mu(x,r)$ 在 $C_{x,r}$ 上服从均匀分布。

我们现在考虑随机变量 $v(x,r)$（表示验证者在与证明者“真正交互”过程中观点的后缀）。回想一下，$v(x,r)$ 是通过均匀选择一个（集合 V_2 的）置换来定义的，并且满足：如果 $v^*(x,r,\pi(G_2))=2$，则 $v(x,r)=(\pi(G_2),\pi)$，否则 $v(x,r)=(\pi(G_2),\pi\circ\phi)$，这里 ϕ 是 G_1 和 G_2 之间的同构映射。显然，对于每个（与 G_2 同构的）H，$v(x,r)=(H,\cdot)$ 的概率等于 $\mathrm{aut}(G_2)/(|V_2|!)$。而且，对于每个（与 G_2 同构的）H，都有

$$\Pr[v(x,r)=(H,\psi)]=\begin{cases}\frac{1}{|V_2|!} & \text{如果 } \psi=\pi\circ\phi^{2-v^*(x,y,H)}\\ 0 & \text{其它}\end{cases}$$

注意到 $H=\psi(G_{v^*(x,y,H)})$ 当且仅当 $\psi=\pi\circ\phi\delta^{2-v^*(x,y,H)}$，我们可推断出 $\mu(x,r)$ 和 $v(x,r)$ 是同分布的。

这样就证明了结论。

同时也完成了命题的第 3 部分的证明。

4.3.3 关于辅助输入的零知识

前面提出的零知识定义不能满足实际应用的要求，因此应该对其稍作修改。我们回想一下，前面提出的零知识定义保证了对于任何公共输入，在与证明者交互后有效计算的结果也能只根据输入而有效计算出来。然而在典型应用中（例如，当一个交互证明被用作较大协议中的子协议的时候），与证明者交互的验证者除了具有公共输入 x，还可能有一些附加的先验信息，用字符串 z 表示，这些先验信息对于验证者从证明者“获取知识”的企图有帮助。这种危险在 z 与 x 有关的情况下变得更为突出（例如，构造 4.3.8 中的协议在验证者有一个输入的两个图之间的同构映射作为先验信息时，就存在严重的危险）。典型的要求是对于任何公共输入 x，在与证明者交互后从 x 和 z 有效计算出来的结果也能只根据 x 和 z 而有效计算出来（不与证明者交互）。这个要求利用定义 4.2.10 中交互证明的扩展概念在下面公式化地描述出来。

定义 4.3.10（重新刻划的零知识）：令 (P,V) 为语言 L 的一个交互证明（如定义 4.2.10 中定义的一样）。记 $P_L(x)$ 为对于 $x\in L$ 满足完备性条件的字符串 y 的集合（即对于每一个 $z\in\{0,1\}^*$，都有 $\Pr[\langle P(y),V(z)\rangle(x)=1]\geq\frac{2}{3}$）。我们说 (P,V) 是关于辅助输入零知识的（或者是辅助输入零知识的），如果对于每个概率多项式时间交互机 V^*，都存在一个运行时间为第一个输入参数长度的多项式的概率算法 M^*，使得如下两个总体是计算不可分辨的（当分辨间距是 $|x|$ 的函数时）：

- $\{\langle P(y_x),V^*(z)\rangle(x)\}_{x\in L,z\in\{0,1\}^*}$，对于任意的 $y_x\in P_L(x)$
- $\{M^*(x,z)\}_{x\in L,z\in\{0,1\}^*}$

即，对于每个运行时间为第一个输入参数长度的多项式的概率算法 D、每个多项式 $p(\cdot)$、所有足够长的 $x\in L$ 以及所有 $y\in P_L(x)$ 和 $z\in\{0,1\}^*$，都有

$$|\Pr[D(x,z,\langle P(y),V^*(z)\rangle(x))=1]-\Pr[D(x,z,M^*(x,z))=1]|<\frac{1}{p(|x|)}$$

在这个定义中，y 表示证明者的先验信息，而 z 表示验证者的先验信息。y 和 z 都可能依赖于公共输入 x；例如，如果 y 促进了证明任务的完成，那么 y 一定依赖 x（例如，当 y 是 $x\in L\in$ **NP** 的一个 **NP** 证据的时候）。我们强调，本地输入（即 y 和 z）对于另一方来说是完全未知的。我

们还要强调，辅助输入 z（而不是 y）也被作为分辨法的输入（分辨算法可看作是扩展的验证者）。

回顾一下，定义 4.2.10 中说的“交互机 V^* 是概率多项式时间的”是指 V^* 的运行时间由公共输入长度的多项式来限定。因此，验证者程序、模拟器和分辨算法都在 x 的长度的多项式时间内（而不是它们所有输入总长度的多项式时间内）运行。这种规定在许多方面都是必要的（除非通过 x 的长度的多项式来明确限定辅助输入的长度；参考习题 11）。例如，允许分辨算法以与辅助输入的长度成正比的时间运行，都可能会使计算零知识退化为完备零知识（例如，考虑一下以公共输入长度的多项式时间运行而又具有长度是公共输入长度的指数函数的巨大辅助输入的验证者）。

定义 4.3.10 涉及到的是计算零知识。具有辅助输入的完备零知识的公式化描述也是显然的。我们指出，在构造 4.3.8 中提出的图同构完备零知识证明事实上是具有辅助输入的完备零知识。这个结论通过对命题 4.3.9 证明中构造的模拟器的稍微扩展就可以得到（即，当激活验证者时，模拟器应该将传送给它的辅助输入提供给验证者）。通常情况下，如果在零知识描述过程中，使用的模拟器通过将验证者程序作为黑盒调用而工作，那么零知识可以通过扩展产生具有辅助输入的零知识。本书中所提到的模拟器都具有这个性质。

高级注释：定义 4.3.10 中隐含的非均匀性

定义 4.3.10 具有非均匀性，这可由分辨器具有辅助输入的事实看出来。事实上这个辅助输入既被送往验证者程序也被送往模拟器，但是如果辅助输入足够长，那么只有分辨器能够利用这个输入的后缀（因为分辨器可能在模拟器的多项式时间界限固定之后才能确定）。这使得定义 4.3.10 中规定的模拟器也通过非均匀多项式规模的电路来产生同真实交互不可分辨的输出。也就是说，对于每个（即使是非均匀的）多项式规模电路族 $\{C_n\}_{n\in\mathbb{N}}$、每个多项式 $p(\cdot)$、所有足够大的 n、所有 $x\in L\cap\{0,1\}^n$ 以及所有 $y\in P_L(x)$ 和 $z\in\{0,1\}^*$，都有

$$|\Pr[C_n(x,z,\langle P(y),V^*(z)\rangle(x))=1]-\Pr[C_n(x,z,M^*(x,z))=1]|<\frac{1}{p(|x|)}$$

下面是这个命题的证明梗概。相反地，我们假定存在一个多项式规模电路族 $\{C_n\}_{n\in\mathbb{N}}$，使得对于无限多的 n，都存在三元组 (x,y,z)，满足 C_n 对于 (x,y,z) 有一个不可忽略的分辨间距。我们通过把对 C_n 的描述与辅助输入 z 合并成一个更长的辅助输入（记为 z'）的方法来导出矛盾。这通过 V^* 和 M^* 都没有足够的时间描述 C_n 来实现。例如，令 $q(\cdot)$ 为限定 V^* 和 M^* 的运行时间的多项式。不失一般性，假设 $|z|\leq q(n)$（否则，z 的其余部分对 V^* 和 M^* 都不可读，都可以忽略）。然后，我们令 Z' 为在 Z 的后面填充空白，并把电路 C_n 的描述放在最后，总长度为 $q(n)$ 的字符串（即，z 是 z' 的前缀）。显然，$M^*(x,z')=M^*(x,z)$ 并且 $\langle P(y),V^*(Z')\rangle(x)=\langle p(y),V^*(z)\rangle(x)$。另一方面，通过使用通用电路估值算法，我们得到一个概率多项式时间算法 D，满足 $D(x,z',\alpha)=C_n(x,z,\alpha)$，这样就得出了矛盾（与 M^* 产生与 (P,V^*) 的输出概率多项式时间不可分的输出的假定矛盾）。

我们指出，定义 4.3.2 自身具有一定的非均匀性，因为它要求对所有的但是有限多的 x 都具有不可分辨性。与此相反，一个该定义完全均匀的模拟将只要求无法找到满足以它作为输入时模拟将失败（关于某个概率多项式时间分辨器）的 x。也就是说，零知识的完全均匀的定义只要求无法找到验证者可以获得知识的 x。更深入的讨论参看 4.4.2.4 节。

高级注释：为什么不寻找一个完全非均匀的公式化描述？

定义 4.3.10 的一个过于简化的版本允许通过(非均匀多项式规模的)电路族模拟验证者和模拟器。假定非均匀电路都采用了辅助输入,因此这一过于简单化的版本就忽略了这些辅助输入。例如,可以要求如下:

对于每个多项式规模的电路族$\{V_n\}_{n\in N}$(表示可能的验证者策略机),都存在一个多项式规模的电路族$\{M_n\}_{n\in N}$(表示模拟器),使得总体$\{\langle P,V_{|x|}\rangle(x)\}_{x\in L}$和$\{M_{|x|}(x)\}_{x\in L}$对于多项式规模的电路来说是不可分辨的。

然而,非均匀电路采用辅助输入的说法是错误的,并且一般情况下,我们发现这些过于简化的定义并不尽如人意。首先,这些定义并不能确保从验证者到模拟器的"有效"转换。实际上,这样的转换在定义 4.3.10 中也不需要,但是在定义 4.3.10 中的对象(即机器)个数是有限的,而在这里我们处理的是无限多个对象(即电路族)。因此,过于简化版本的定义所提供的"安全"级别是不能令人满意的。其次,过于简化版本的定义也无法确保验证者非均匀部分的大小与模拟器相应部分的大小之间的关系,然而在定义 4.3.10 中,惟一的非均匀部分是辅助输入,这部分是保持不变的。当要证明非常数次重复零知识证明系统的顺序组合定理时也会出现这两个问题。最后,我们要注意,这个简化版本的定义并未隐含基本版本的定义(即定义 4.3.2);例如,考虑对于公共输入 x,发送只与$|x|$有关的难于计算(例如,在区间$[2^{|x|}+1,\ldots,2^{|x|}+|x|^3]$内所有整数的素因数分解)的$|x|$比特长字符串的验证者。

4.3.4 零知识证明的顺序合成

零知识证明的定义必须满足的一个直观要求是,零知识证明在顺序合成下是封闭的。也就是说,如果我们在执行完一个零知识证明后执行另一个零知识证明,那么合成执行的结果也一定是零知识的。同样的结果对于一个接一个地执行多项式个零知识证明也应该成立。事实上,我们将会看到修订的零知识定义(即定义 4.3.10)满足这个要求。有趣的是,定义 4.3.2 中定义的零知识证明在顺序合成下不满足封闭性,这实际上也是我们要扩展这个定义的原因(在定义 4.3.10 中进行了扩展)。

除了概念上的重要性之外,顺序合成引理在零知识证明系统的设计中也是一个重要的工具。典型地,这样的证明系统包含许多原子零知识证明的多次重复。不严格地说,原子证明为结论的有效性提供了一些(但不是很多)统计证据。重复这一原子证明足够多次可增加对该结论有效性的信心。更准确地说,原子证明提供了对于来自语言内部和外部的字符串在接受概率上的差别。例如,在构造 4.3.8 中,同构图对(即来自 GI 的输入)被接受的概率为 1,而非同构图对(即 GI 以外的输入)被接受的概率至多为$\frac{1}{2}$。通过重复原子证明,两种概率之间的差别越来越明显。例如,重复构造 4.3.8 中的证明 k 次,将会产生一个新的交互证明,在这个新的交互证明中,接受 GI 内输入的概率仍然为 1,而接受 GI 外输入的概率至多为$\frac{1}{2^k}$。顺序合成引理确保了如果原子证明系统是零知识的,那么重复多项式次原子证明系统所生成的证明系统也是零知识的。

在给出顺序合成引理之前,我们提醒读者注意:交互证明的零知识特性实际上就是证明者的零知识特性。而且,要求证明者仅仅对语言内的输入是零知识的。最后,我们强调,当讨论关于辅助输入的零知识时,我们指的是验证者所有可能的辅助输入。

引理 4.3.11(顺序合成引理):令 P 是一个关于某种语言 L 内辅助输入的零知识交互机(即证明者)。假定在输入 x 下,P 发送的最后一条消息是一个特殊的证明结束符。令 $Q(\cdot)$ 是一个多项式,P_Q 是一个在公共输入 x 下以 $Q(|x|)$ 个阶段运行的交互机,每个阶段都包含公共输入 x 下 P 的运行(我们强调,因为 P 是概率的,所以交互机 P_Q 对于 $Q(|x|)$ 中的每个阶段都采用独立的硬币投掷机制)。那么,P_Q(关于辅助输入)在 L 上是零知识的。而且,如果 P(关于辅助输入)是完备零知识的,那么 P_Q(关于辅助输入)也是完备零知识的。

证明结束符的约定是出于技术目的而引入的(这种约定对于所有已知的证明系统都是多余的,对能够从公共输入的长度计算出已经发送消息的数量的证明系统也是多余的)。显然,每个机器 P 都可以通过调整使得最后一条消息是一个恰当的符号(如前面假定的那样),并且这样做仍然能保持 P 具有零知识特性(以及完备性条件和合理性条件)。

这个引理忽略了重复交互证明若干次的其他方面,特别地,忽略了接收语言内输入的概率和拒绝语言外输入的概率之间间距的影响。这种重复交互证明系统带来的影响在 4.2.1.3 节中有所讨论(也可以参考习题 1)。

证明:令 V^* 为任意一个同组合证明者 P_Q 交互的概率多项式时间交互机。我们的任务是构造一个用来模拟 V^* 和 P_Q 的真实交互的(多项式时间)模拟器 M^*。下面是对这一模拟的详尽说明。关键的思想是模拟在共同输入 x 下,与 P_Q 的操作阶段相对应的 $Q(|x|)$ 个阶段的真实交互。P_Q 操作的每个阶段都通过原子证明者 P 的模拟器来模拟。验证者在每个阶段中积累的信息都通过辅助输入传到下一阶段。

(在随后的说明中,我们忽略证明者的辅助输入。这仅仅是为了简化我们的符号。也就是说,我们写的不是 $P(y)$ 和 $P_Q(y)$,而是 P 和 P_Q,此处 y 是证明者的辅助输入。)

执行上面模拟的第一步是将任意交互机 V^* 的执行过程分割成几个阶段。这个分割过程可能不是在程序 V^* 的代码中实现的,但是却可以影响程序 V^* 的执行过程。分割过程通过指定证明者 P_Q 的阶段结构来实现,阶段结构是通过证明结束符来分辨的。因此,我们得出结论:无论 V^* 如何操作,在公共输入 x 下,V^* 和 P_Q 的交互可由连续 $Q(|x|)$ 次相关机器(记为 V^{**})与 P 的交互来实现。

结论 4.3.11.1:存在一个概率多项式时间的 V^{**},使得对于每个公共输入 x 和辅助输入 z,都有

$$\langle P_Q, V^*(z)\rangle(x) = Z^{(Q(|x|))}$$

其中 $Z^{(0)} \stackrel{\text{def}}{=} z, Z^{(i+1)} \stackrel{\text{def}}{=} \langle P, V^{**}(Z^{(i)})\rangle(x), i=0,\ldots,Q(|x|)-1$。

也就是说,$Z^{(Q(|x|))}$ 是描述在公共输入 x 下,V^{**} 与 P 连续 $Q(|x|)$ 次交互后输出的随机变量,其中 V^{**} 在第 $i+1$ 次交互后的辅助输入等于 V^{**} 在第 i 次交互(即,$Z^{(i)}$)后的输出。

证明:直观地讲,V^{**} 捕获了 V^* 的每一个阶段的功能。在一个阶段中的功能,我们指的是 V^* 将这个阶段开始时工作磁带的内容转换为这个阶段结束时工作磁带的内容的方式和在这个阶段中 V^* 确定它发送的消息的方式。实际上,这种转换依赖于当前阶段收到的消息。我们强调,与其说我们通过"反编译"V^*(的代码)来影响这个转换,还不如说是通过监控 V^* 的所有磁带并模拟它的执行来影响这个转换。详细说明如下。

为了使描述过程更清楚,我们首先修改 V^*,使其所有的"基本"活动仅与它的工作磁带相关。我们将对 V^* 稍作修改,使得 V^* 在开始执行时将公共输入、随机输入和辅助输入读入到工作磁带的特定区域中,并且再也不能访问这个只读磁带。同样地,我们还将修改 V^*,使其

在活动期[①]（参见定义 4.2.2）开始时从通信磁带中将输入的消息读入到工作磁带的特定区域中（并且在这个活动期内不能再访问输入消息磁带）。实际上，应该修改这一描述，使得 V^*（在公共输入下）只复制这些磁带中每一个上面内容的多项式长的前缀，这个多项式就是用来限定（没有修改前的）V^* 的运行时间的多项式。

（说得正式一些，给定任意一个 V^*，我们构造一个以满足前述条件的方式模拟 V^* 的机器 W^*；也就是说，即使 V^* 不满足这些条件，W^* 也会满足这些条件。机器 W^* 会有一些额外的工作磁带，这些磁带将被指定作为 V^* 的公共输入、随机输入和输入通信磁带。机器 W^* 会通过将自己的公共输入、随机输入和辅助输入复制到相对应的指定磁带中去来启动。同样地，机器 W^* 可以通过从自己的通信磁带中复制当前输入消息到相对应的指定磁带（即 V^* 的输入通信磁带）中来启动每个活动期。在完成这些复制操作之后，我们就来模拟 V^* 的执行。显然，V^* 满足这些指定的要求。这样，可以正式地说，以后无论什么时候谈到 V^*，我们指的都是 W^*。）

考虑对于公共输入 x，$V^*(z)$ 与 P_Q 的交互过程。通过前面的修改，交互包括 $Q(|x|)$ 个阶段，其中除了在第一个阶段外的所有阶段中，机器 V^* 都不访问它的公共输入、随机输入和辅助输入磁带。（在第一个阶段中，机器 V^* 通过将这些磁带的内容复制到它的工作磁带上来启动，并且以后再也不访问这些磁带。）同样地，在执行当前阶段时，机器 V^* 不会从它的输入磁带中读取前一阶段的消息（但是，它可能会从它的工作磁带存储的信息中读取这些"旧的"信息）。考虑一下 V^* 在 $Q(|x|)$ 个阶段的每个阶段结束时工作磁带的内容，自然就得出了 V^{**} 的构造。

我们现在已经为提出 V^{**} 的构造做好了最后的准备：对于公共输入 x 和辅助输入 z'，机器 V^{**} 通过将 z' 复制到工作磁带开始活动。然后，机器 V^{**} 以上面 V^{**} 的工作磁带的内容（而不是以空的工作磁带的内容）开始模拟 V^{**} 与 P_Q（对于输入 x）的交互的一个阶段。被模拟的机器 V^* 把机器 V^{**} 的通信磁带当作它自己的通信磁带。当 V^* 完成了当前阶段的交互时，V^{**} 输出 V^* 的工作磁带的当前内容，然后中止运行。因此，当 z' 等于 V^* 在 $i \geq 1$ 阶段后工作磁带上的可能内容时，被模拟的 V^* 应该按照第 $i+1$ 阶段运行，并且 V^{**} 的输出与第 $i+1$ 阶段后 V^* 工作磁带的内容同分布。实际上，为了处理和 P_Q 交互的第一个阶段，前面的叙述应略作改动（即，前面忽略了 $i=0$ 的情形）。特别地，V^{**} 仅当 z' 对 V^* 工作磁带的内容进行编码的时候才复制 z' 到 V^* 的工作磁带中（不失一般性，我们假定 V^* 工作磁带内容的编码与 V^* 辅助输入的编码是不同的）。在 z' 将辅助输入编码进 V^* 的情况下，V^{**} 在空白工作磁带上调用 V^*，并且 V^* 将 V^{**} 的可读磁带（即公共输入磁带、随机输入磁带和辅助输入磁带）也看作是自己的磁带。注意，$Z^{(1)} \stackrel{\text{def}}{=} \langle P, V^{**}(z)\rangle(x)$ 表示在（对于公共输入 x 和辅助输入 z 与 P_Q 交互的）第一个阶段之后 V^* 工作磁带的内容。同样地，对于每个 $i=2,\ldots,Q(|x|)$，随机变量 $Z^{(i)} \stackrel{\text{def}}{=} \langle P, V^{**}(Z^{(i-1)})\rangle(x)$ 表示在第 i 个阶段后 V^* 工作磁带的内容。这样就证明了结论。■

因为 V^{**} 是一个与 P 交互的（具有辅助输入的）多项式时间交互机，所以由引理可知，存在一个能够在以第一个输入长度为变量的多项式时间内模拟这些交互的概率机。用 M^{**} 表示这个模拟器[②]。对于每个（以 x 为变量的）概率多项式时间算法 D、每个多项式 $p(\cdot)$、所有

---

① 注意，交互机 M 执行期间的活动期包括从接收到最后一条消息的时间到 M 发送它的响应消息的时间之间 M 所采取的所有步骤。

② 回想在完备零知识的情况中（参见定义 4.3.1），机器 M^{**} 可能没有真实的输出就中止了（而不是以输出⊥来中止）。然而，通过足够多次重复，我们能够使这一事件的概率以指数级降低。在这段说明的剩余部分，为了简化，我们假设 M^{**} 总是以输出来中止运行。

足够长的 $x\in L$，以及所有的 $z\in\{0,1\}^*$，我们都有

$$|\Pr[D(x,z,(P,V^{**}(z))(x))=1]-\Pr[D(x,z,M^{**}(x,z))=1]|<\frac{1}{p(|x|)}\quad(4.1)$$

我们现在已经为构造模拟与 P_Q 交互后 V^* 的"真实"输出的模拟器 M^* 做好了准备。不失一般性，我们可以假定 V^* 的输出等于交互结束时它的工作磁带的内容(因为这时 V^* 的输出可以从它的工作磁带的内容以概率多项式时间计算出来)。机器 M^* 使用模拟器 M^{**}(作为黑盒)。

模拟器 M^*：在输入为(x,z)的情况下，机器 M^* 设 $z^{(0)}=z$，并且运行 $Q(|x|)$个阶段。在第 i 个阶段中，机器 M^* 通过以$(x,z^{(i-1)})$作为输入来运行 M^{**} 从而计算出 $z^{(i)}$。在完成 $Q(|x|)$个阶段后，M^* 以输出 $z^{(Q(|x|))}$来停止运行。

很明显，这里构造的机器 M^* 在它的第一个输入的多项式时间内运行。剩下的是要证明机器 M^* 实际上产生了 V^*(与 P_Q 交互后)的输出多项式不可分辨的输出。

结论 4.3.11.2：对于每个运行时间为第一个输入的多项式的概率算法 D、每个多项式 $p(\cdot)$、所有足够长的 $x\in L$ 和所有的 $z\in\{0,1\}^*$，我们都有

$$|\Pr[D(x,z,(P_Q,V^*(z))(x))=1]-\Pr[D(x,z,M^*(x,z))=1]|<\frac{1}{p(|x|)}$$

而且，如果 P 是完备零知识的，那么$(P_Q,V^*(z))(x)$和 $M^*(x,z)$是同分布的。

略证：我们使用混合理论的方法(参考第3章)进行证明。特别地，我们定义如下 $Q(|x|)+1$ 个混合分量。第 i 个混合分量，$0\leq i\leq Q(|x|)$，对应于下面的随机化过程。我们首先令 V^{**} 以公共输入 x 和辅助输入 z 开始，同 P 交互 i 个阶段，并且记 V^{**} 在第 i 个阶段后的输出为 $Z^{(i)}$，然后我们重复迭代 M^{**} $Q(|x|)-i$ 次。在这两种情况下，我们都使用前一阶段的输出作为后一阶段的辅助输入。形式上，混合分量 $H(i)$定义如下。

$$H^{(i)}(x,z)\stackrel{\text{def}}{=}M^{**}_{Q(|x|)-i}(x,Z^{(i)})$$

其中，$Z^{(i)}$与结论 4.3.11.1 中定义的相同，$M_0^{**}(x,z')\stackrel{\text{def}}{=}z'$，$M_k^{**}(x,z')\stackrel{\text{def}}{=}M_{k-1}^{**}(x,M^{**}(x,z'))$，对于 $k=1,\ldots,Q(|x|)-i$。

由结论 4.3.11.1，第 $Q(|x|)$个混合分量(即 $H^{(Q(|x|))}(x,z)=Z^{(Q(|x|))}$)等于$(P_Q,V^*(z))(x)$。另一方面，回想 M^* 的构造，我们可知第 0 个混合分量(即 $H^{(0)}(x,z)=M^{**}_{Q(|x|)}(x,z)$)等于 $M^*(x,z)$。这样，为了完成证明，我们只需要证明任何两个相邻的混合分量都是计算不可分辨的(因为这意味着端点混合分量 $H^{(Q(|x|))}$和 $H^{(0)}$也是不可分辨的)。为此，我们将第 i 个和第 $i-1$ 个混合分量改写如下：

$$\begin{aligned}H^{(i)}(x,z)&=M^{**}_{Q(|x|)-i}(x,Z^{(i)})\\&=M^{**}_{Q(|x|)-i}(x,(P,V^{**}(Z^{(i-1)}))(x))\\H^{(i-1)}(x,z)&=M^{**}_{Q(|x|)-(i-1)}(x,Z^{(i-1)})\\&=M^{**}_{Q(|x|)-i}(x,M^{**}(x,Z^{(i-1)}))\end{aligned}$$

其中 $Z^{(i-1)}$的定义与结论 4.3.11.1 中定义的相同。

通过使用平均理论的方法，如果算法 D 可分辨混合分量 $H^{(i)}(x,z)$和 $H^{(i-1)}(x,z)$，那么就存在一个 z'(在 $Z^{(i-1)}$的支集里)使得 D 也能分辨随机变量 $M^{**}_{Q(|x|)-i}(x,(P,V^{**}(z'))(x))$和 $M^{**}_{Q(|x|)-i}(x,M^{**}(x,z'))$。(在前面两种情况下，$D$ 都具有 x 和 z 作为输入。)通过算法 M^{**} 和 D，我们可得到一个新的算法 D'，是运行时间为多项式的算法，算法也能分辨随机变量

$(x,z,i,z',\langle P,V^{**}(z')\rangle(x))$和$(x,z,i,z',M^{**}(x,z'))$。特别地，在输入$(x,(z,i,z')\alpha)$下(这里的$\alpha$取自$\langle P,V^{**}(z')\rangle(x)$或者$M^{**}(x,z')$)，算法$D'$调用输入为$(x,z,M^{*}_{Q(|x|)-i}(x,\alpha))$的算法$D$，并且不论$D$得出什么结果都将其输出。显然有：

$$|\Pr[D'(x,(z,i,z'),\langle P,V^{**}(z')\rangle(x))=1]-\Pr[D'(x,(z,i,z'),M^{**}(x,z'))=1]|$$
$$\geq|\Pr[D(x,z,H^{(i)}(x,z))=1]-\Pr[D(x,z,H^{(i-1)}(x,z))=1]|$$

注意，D'使用了额外输入(x,z,i,z')，但是能够分辨$\langle P,V^{**}(z')\rangle(x)$和$M^{**}(x,z')$。这与(辅助输入)零知识的分辨器定义并不相符，因为该分辨器只接受(x,z')和待分辨的字符串。换句话说，我们实际上已经构造了一个$D'=D'_{i,z}$，它依赖于i和z来分辨$\langle P,V^{**}(z')\rangle(x)$和$M^{**}(x,z')$。而且，在完备零知识的情况下，令$D$为任意函数(而不是一个有效算法)，这已经足够反驳$M^{**}$完美地模拟了$(P,V^{**})$的假设。在计算零知识的情况下，我们用到这样的事实，即辅助输入零知识的定义隐含了抗非均匀(多项式大小的)分辨器的稳定性，并且我们注意到$D'_{i,z}$恰好是这样的分辨器(因为D就是这样的分辨器)。这样，在这两种情况下，都导致了矛盾(与M^{**}模拟(P,V^{**})的假设矛盾)■。

结论 4.3.11.2 证明的更多细节：现在(假定读者已经阅读过第 3 章)，读者应该能将前面的证明要点转化为一个详细的证明。证明要点中欠缺的主要是从与结论 4.3.11.2 的论述矛盾的算法推导出与“M^{**}是(P,V^{**})的模拟器”的假设矛盾的算法的细节。这些细节在下面讲述。

相反地，我们假设存在一个概率多项式时间算法D和一个多项式$p(\cdot)$，满足对于无穷多的$x\in L$，都存在$z\in\{0,1\}^*$使得：

$$|\Pr[D(x,z,\langle P_Q,V^*(z)\rangle(x))=1]-\Pr[D(x,z,M^*(x,z))=1]|>\frac{1}{p(|x|)}$$

因此，对于每一个这样的x和z，都存在一个$i\in\{1,\ldots,Q(|x|)\}$满足

$$|\Pr[D(x,z,H^{(i)}(x,z))=1]-\Pr[D(x,z,H^{(i-1)}(x,z))=1]|>\frac{1}{Q(|x|)\cdot p(|x|)}$$

这里混合分量$H(j)$与前面定义的相同。记$\varepsilon(n)\stackrel{\text{def}}{=}1/Q(n)\cdot p(n))$。就像以前一样，用一个平均理论将第$i$个和第$i-1$个混合分量组合起来，于是得到，对于每个这样的$x$，$z$和$i$，都存在一个$z'$，满足

$$|\Pr[D(x,z',M^{*}_{Q(|x|)-i}(x,\langle P,V^{**}(Z')\rangle(x)))=1]$$
$$-\Pr[D(x,z',M^{*}_{Q(|x|)-i}(x,M^{**}(x,z')))=1]|>\varepsilon(|x|)$$

这已经快推导出我们想要的矛盾了。也就是说，如果我们“知道”i和z，随机变量$(x,z'\langle P,V^{**}(z')\rangle(x))$和$(x,z',M^{**}(x,z'))$就可以用算法$D$和$M^{**}$分辨出来。但是我们怎样才能“知道”$i$和$z$呢？这个问题可以通过前面指出的事实进行解决，这个事实就是：即使使用非均匀多项式规模的电路，也无法分辨M^{**}的输出和V^{**}与P的交互。因此，为了得出矛盾，构造一个将i和z整合进它的说明的非均匀分辨器就足够了。另一种可选的方式是，我们可将i和z整合为一个新的辅助输入，记其为z''，使得z'是z''的一个前缀，但是在V^*和M^*看来，z''和z'一样。接下来我们来谈谈这种可选方式，具体说明如下。

令T表示V^*和M^*的时间复杂度的多项式上界。注意，和先前一样，对于每个由(x,z)决定的z'，必须满足$|z'|\leq T(|x|)$(因为z'是$M^*(x,z)$的局部计算的一个可能记录)。令$z''=(z'b^{T(|x|)-|z'|}i,z,)$，其中$i$和$z$和以前定义的一样($b$表示工作磁带的空白符)。我们构造

一个可分辨关于多元组(x,z,i,z')的$(x,z'',(P,V^{**}(z''))(x))$和$(x,z'',M^{**}(x,z''))$的概率多项式时间算法 D'。在输入(x,z'',α)下(这里假设 α 在$\langle P,V^{**}(z'')\rangle(x)=\langle P,V^{**}(z')\rangle(x)$中,或者在 $M^{**}(x,z'')=M^{**}(x,z')$中),算法 D' 首先由 z''求出 i 和 z'来。然后,算法 D' 使用 M^{**} 计算 $\beta=M^{**}_{Q(|x|)-i}(x,\alpha)$。最后,$D'$输出 $D(x,z,\beta)$,并中止运行。由 V^{**} 和 M^{**} 都无法分辨辅助输入 z'和 z''的事实,我们有

$$
\begin{aligned}
&|\mathrm{Rr}[D'(x,z'',\langle P,V^{**}(z'')\rangle(x)) = 1] - \Pr[D'(x,z'',M^{**}(x,z'')) = 1]| \\
&\quad = |\mathrm{Rr}[D'(x,z'',\langle P,V^{**}(z')\rangle(x)) = 1] - \mathrm{Rr}[D'(x,z'',M^{**}(x,z')) = 1]| \\
&\quad = |\Pr[D(x,z,M_{Q(|x|)-i}(x,\langle P,V^{**}(z')\rangle(x))) - 1] \\
&\qquad - \Pr[D(x,z,M_{Q(|x|)-i}(x,(M^{**}(x,z')))) = 1]| \\
&\qquad\quad > \epsilon(|x|)
\end{aligned}
$$

这样就得到了矛盾(与 M^{**} 是(P,V^{**})的模拟器的假设矛盾)。■

引理得证。

考虑并行合成怎样?

不幸的是,我们无法证明在并行合成运算下,零知识(即使是关于辅助输入的零知识)特性仍然保持。而且,确实存在这样的(辅助输入)零知识,在并行执行两次后就泄漏(给"恶意验证者")了知识。更深入的细节参见 4.5.4 小节。

在协议并行合成运算下,零知识特性无法保持确实不是一件好事。有人可能会说这在概念上就是一个令人烦恼的现象,我们不同意这种说法。我们的想法是,一般来说(即,不仅仅局限于零知识的范畴),并行合成运算下协议和"规则"的动作要比顺序合成运算下的动作和规则复杂得多;实际上,在其他一些情况下(例如,计算合理证明、知识的证明和多证明者证明系统;分别参见 4.8、4.7 和 4.11 节),并行合成要滞后于顺序合成。而且,相对于顺序合成来说,并行合成的惟一优点在于其效率。这样,我们并未将并行合成运算下的不封闭性看作是零知识公式的重要缺陷。然而零知识的"不封闭性"引发了在并行合成运算下仍然保持的可选(相关)概念的研究(这种概念可能或强于或弱于零知识的定义)。更深入的细节参见 4.9 节和 4.6 节。

4.4　*NP* 零知识证明

这一节是本章的核心内容,也就是说,我们将会讲解为 *NP* 中每种语言建立零知识证明的方法。这种方法的重要性在于它的通用性,而通用性正是许多应用的关键。特别地,在实际中想要证明的所有结论几乎都可以归结为证明是否是 *NP* 中语言的成员的问题。特别地,这样陈述的零知识证明的构造提供了一个"驱使"用户("forcing"party)正确执行给定协议的工具。

构造 *NP* 语言的零知识证明的方法使用了比特承诺的概念。因此,我们先来给出比特承诺的概念。(建议在学习比特承诺方案之前想了解更多比特承诺方案的应用的读者,可以先阅读 4.4.2.1 节。)

4.4.1 承诺方案

承诺方案是许多密码协议的基础成分。通过使用承诺方案可以使一方将自己承诺为一个特定的值并保持这个值的保密性。在后面的阶段中承诺“被打开”，并且“打开”产生一个在承诺阶段就确定的值。承诺方案是不透明信封的数字对等体。通过将注记(note)放进这样的信封中，一方就可以承诺它放入的注记的内容而又保持内容的保密性。

4.4.1.1 定义

不严格地说，承诺方案是一个有效的两阶段双方协议，通过这个协议，称为发送者的一方可以将自己承诺为一个值，满足下面两个相互冲突的要求。

1. 保密性(或者隐藏性)：在第一个阶段结束时，称为接收者的一方不会得到任何有关发送者的值的知识。这个要求甚至在接收者试图欺骗的时候也一定要满足。

2. 明确性(或者绑定性)：在第一个阶段中给定交互副本的时候，最多只存在一个值能够使接收者将这个值作为承诺的合法“公开”来接收。这个要求甚至在发送者试图欺骗的时候也一定要满足。

此外，协议应该是可行的，意思就是说如果双方都遵守它，那么在第二个阶段结束时接收者应该得到了由发送者承诺的有效值。第一个阶段称为承诺阶段，第二个阶段称为公布阶段。我们需要在承诺阶段不泄漏任何知识(至少是有关发送者的值的知识)给接收者，并且在承诺阶段确实将发送者“绑定”到一个惟一的数值上(也就是在公布阶段，接收者只可以接收这个值)。我们强调，协议应该是有效的，即双方的预定程序可以在概率多项式时间内执行。不失一般性，公布阶段可以只包括发送者将原始数值和在承诺阶段运用的随机硬币投掷的序列发送给接收者。当且仅当提供的信息与它在承诺阶段中的交互副本匹配时，接收者才会接收。后面的约定引出如下的定义(仅对于承诺阶段)。

定义 4.4.1(比特承诺方案)：**比特承诺**方案是一对概率多项式时间交互机(S,R)(对应于发送者和接收者)，满足如下条件：

- 输入规定：公共输入是一个一元整数 n(作为安全参数)。
- 发送者的私人输入是一个比特，记做 v。
- 保密性(或隐藏性)：接收者(无论怎样背离协议都)不可以区分 0 承诺和 1 承诺。也就是说，对于每个与 S 交互的概率多项式时间机器 R^*，在$\{\langle S(0),R^*\rangle(1^n)\}_{n\in N}$和$\{\langle S(1),R^*\rangle(1^n)\}_{n\in N}$这两种情况下，描述 R^* 输出的概率总体是计算不可分辨的。
- 明确性(或绑定性)：这个要求的预备知识：

1. 接收者对与发送者之间交互的观点，记为$(r,\overline{m})$，包括接收者(r)使用的随机硬币和从接收者$(\overline{m})$接收到的消息序列。

2. 令 $\sigma\in\{0,1\}$。我们说，接收者(对这样的交互)的观点$(r,\overline{m})$是**或然 σ 承诺**如果存在一个字符串 s，满足 $\overline{m}$ 描述当 R 使用本地硬币 r，并且与使用本地硬币 s 并具有输入$(\sigma,1^n)$的机器 S 交互时，从 R 接收到的消息。

(使用定义 4.3.3 的符号，如果$(r,\overline{m})=\text{view}_{R(1^n,r)}^{S(\sigma,1^n,s)}$，我们就说$(r,\overline{m})$是或然 σ 承诺。)

3. 我们说接收者的观点是不明确的$(r,\overline{m})$，如果它既是或然 0 承诺又是或然 1 承诺。

明确性要求说明，对于所有的除了可以忽略的一小部分接收者的硬币投掷外的所有硬币投掷，都不存在（来自发送者的）消息序列与这些硬币投掷形成不明确的接收者观点。也就是说，除了可以忽略的 $r\in\{0,1\}^{\mathrm{poly}(n)}$，在其他任何情况下，都不存在 $\overline{m}$，满足$(r,\overline{m})$是不明确的。

保密性要求是指计算保密性。另一方面，明确性要求具有信息理论的成分（即，并不是指计算能力上的明确性要求），有时也称为完备的（或者绝对的）。因此，定义 4.4.1 中的承诺方案有时也叫做计算隐藏和完备绑定的。一个要求信息论保密性和产生不明确性的计算不可行性的对偶定义将在 4.8.2 小节中给出（这个定义也叫做完备隐藏和计算绑定的）。

典型的公布阶段　保密性的要求明指在承诺阶段完成时的要求。另一方面，我们强调，明确性要求暗指公布阶段采用如下形式：

1. 发送者发送给接收者的是，它的私有输入 v 和它在承诺阶段中使用的随机硬币 s。

2. 接收者要进行验证的是，v 和 s（也包括 R 在承诺阶段所使用的硬币(r)）事实上是否能够产生 R 在承诺阶段收到的消息。验证是在多项式时间内进行的（通过运行程序 S 和 R）。

注意，实用性要求（也就是说，如果双方都遵循协议，那么在公布阶段结束的时候接收者应该得到 v）在这个约定中隐含地满足。

4.4.1.2　基于单向置换的构造

某些公钥加密方案可以被用来作为承诺方案。这可以通过让发送者产生一对密钥，并将公钥和对某个值加密后的结果作为它对这个值的承诺来实现。为了满足明确性的要求，选用的公钥方案需要满足一些附加的要求（即，合法的公钥集合应该是容易识别的，而且每一个用合法公钥进行的加密都只有惟一的解密）。在某些场合下，公钥加密方案具有一些承诺方案并没有要求的附加特性，这些特性的存在似乎需要更强的难驾驭的假设（因此，我们认为前述的方法是"概念性错误的"）。接下来要提到的可选构造使用了单向置换。特别地，我们使用了一个单向置换，记为 f，和一个对它的核心断言，记为 b（参考 2.5 节）。事实上，我们可以使用任何 1-1 的单向函数。

构造 4.4.2（简单比特承诺）：令 f 是一个 $\{0,1\}^*\rightarrow\{0,1\}^*$ 的函数，b 是一个 $\{0,1\}^*\rightarrow\{0,1\}$ 的断言。

1. 承诺阶段：为了承诺为值 $v\in\{0,1\}$（使用安全参数 n），发送者均匀地选择 $s\in\{0,1\}^n$，并将数对$(f(s),b(s)\oplus v)$发送给接收者。

2. （规范的）公布阶段：在公布阶段中，发送者公布比特 v 和在承诺阶段使用的字符串 s。接收者接受值 v，如果 $f(s)=\alpha$ 并且 $b(s)\oplus v=\sigma$，其中(α,σ)是接收者对承诺阶段的观点。

命题 4.4.3：令 $f:\{0,1\}^*\rightarrow\{0,1\}^*$ 是一个 1-1 的单向函数，$b:\{0,1\}^*\rightarrow\{0,1\}$是对 f 的绝对判定，那么构造 4.4.2 中提出的协议就构成了一个比特承诺方案。

证明：保密性的要求直接从 b 是 f 的核心断言这个事实可以得到。实际上，不存在不明确的接收者观点。也就是说，对于每个可能的接收者观点(α,σ)，都存在一个惟一的 $s\in\{0,1\}^{|\alpha|}$ 使得 $f(s)=\alpha$ 成立，因此也存在一个惟一的 $v\in\{0,1\}$使得 $b(s)\oplus v=\sigma$ 成立。

4.4.1.3　基于单向函数的构造

我们现在基于最弱的可能假定：单向函数的存在性给出一种比特承诺方案的构造。证明这个假定的要求是最低的留作习题（即习题 13）。另一方面，运用第 3 章的结果（定理 3.3.3

和 3.5.12)可知，存在单向函数就意味着存在可以将 n 位字符串扩展为 $3n$ 位字符串的伪随机发生器。我们将会在下面的结构中使用这样的伪随机发生器。

我们从构造的动机开始描述。令 G 是一个满足 $|G(s)|=3\cdot|s|$ 的伪随机发生器。假定 G 具有如下特性：集合 $\{G(s):s\in\{0,1\}^n\}$ 和 $\{G(s)\oplus 1^{3n}:s\in\{0,1\}^n\}$ 不相交，其中 $\alpha\oplus\beta$ 表示字符串 α 和 β 的异或(XOR)。发送者通过均匀选择 $s\in\{0,1\}^n$ 来将自己承诺为比特 v，并发送消息 $G(s)\oplus v^{3n}$(v^k 表示所有比特都取值为 v 的 k 比特长的字符串)。不幸的是，前述的假设在一般的情况下不成立，需要进行一些稍微复杂的修改。注意，对于大部分字符串 $r\in\{0,1\}^{3n}$，集合 $\{G(s):s\in\{0,1\}^n\}$ 和 $\{G(s)\oplus r:s\in\{0,1\}^n\}$ 是不相交的。这样的字符串 r 称为好的。从这一点提出了如下协议：接收者均匀选择 $r\in\{0,1\}^{3n}$，希望它是好的，并将 r 发送给发送者。收到 r 之后，发送者通过均匀选择 $s\in\{0,1\}^n$ 承诺为比特 v，并在 $v=0$ 时发送消息 $G(s)$，否则发送 $G(s)\oplus r$。

构造 4.4.4(一般假定下的比特承诺)：令 $G:\{0,1\}^*\rightarrow\{0,1\}^*$ 是一个函数，满足对于所有的 $s\in\{0,1\}^*$，都有 $|G(s)|=3\cdot|s|$。

1. 承诺阶段：

- 为了接受一个比特的承诺(使用安全参数 n)，接收者均匀选择 $r\in\{0,1\}^{3n}$，并将它发送给发送者。
- 当(从接收者那里)接收到消息 r 时，接收者通过均匀选择 $s\in\{0,1\}^n$ 来承诺值 $v\in\{0,1\}$，并且如果 $v=0$，它就发送消息 $G(s)$，否则发送 $G(s)\oplus r$。

2. (规范)公布阶段：在公布阶段，发送者公布在承诺阶段使用的字符串 s。如果 $G(s)=\alpha$，接收者接受值 0；如果 $G(s)\oplus r=\alpha$，接受值 1；其中 (r,α) 是接收者对承诺阶段的观点。

这种(规范)公布阶段的定义允许接收者接受两个值，但是我们将会说明这种情况是很少出现的(甚至不出现)。

命题 4.4.5：如果 G 是一个伪随机发生器，那么构造 4.4.4 中提出的协议构成一个比特承诺方案。

证明：保密性的要求从 G 是一个伪随机发生器的事实可以得到。特别地，令 U_k 表示在长度为 k 的字符串上均匀分布的随机变量。那么，对于每个 $r\in\{0,1\}^{3n}$，随机变量 U_{3n} 和 $U_{3n}\oplus r$ 是同分布的。因此，如果能够找到一个 $r\in\{0,1\}^{3n}$，使得 $G(U_n)$ 和 $G(U_n)\oplus r$ 是计算可分辨的，那么 U_{3n} 和 $G(U_n)$ 是计算可分辨的，或者 $U_{3n}\oplus r$ 和 $G(U_n)\oplus r$ 是计算可分辨的。在每种情况下，都能得到与 G 的伪随机性相矛盾的结果。

我们现在来考虑明确性要求。按照这个构造的动机性讨论中的表示方法，我们称 $r\in\{0,1\}^{3n}$ 是好的，如果集合 $\{G(s):s\in\{0,1\}^n\}$ 和 $\{G(s)\oplus r:s\in\{0,1\}^n\}$ 是不相交的。我们说 $r\in\{0,1\}^{3n}$ 在种子 s_1 和 s_2 之间产生了冲突，如果 $G(s_1)=G(s_2)\oplus r$。显然，如果 r 不在任何一对种子之间产生冲突的话，那么 r 就是好的。另一方面，对给定的一对种子 (s_1,s_2)，最多有一个字符串 r 可以使它们产生冲突；也就是说，$r=G(s_1)\oplus G(s_2)$。因为最多可能有 $\binom{2^n}{2}<2^{2n}$ 对种子，少于 2^{2n} 的字符串将会在种子对之间产生冲突，因此所有其他的 $3n$ 比特长的字符串都是好的。这样就得出，接收者至少以 $1-2^{2n-3n}$ 的概率选择了好的字符串，在这种情况下接收者的观点 (r,α) 是明确的(因为如果 r 是好的，$G(s_1)=\alpha$ 对某个 s_1 成立，那么 $G(s_2)\neq\alpha\oplus r$ 必须对所有的 s_2 都成立)。明确性要求得证。■

4.4.1.4　扩展

比特承诺方案的定义和结构可以很容易地扩展到一般的承诺方案，使得发送者可以承诺一个字符串而不仅仅是一个比特。事实上，为了本节后面的叙述，我们需要一个通过它可以承诺三进制数值的承诺方案。对这种特殊情形进行处理的定义和构造的扩展也是非常直接的。

在本节的其余部分中，我们将需要一个比前面定义的比特承诺方案具有更强保密性要求的承诺方案。特别地，我们不考虑关于多项式时间机器的保密性要求，而是考虑关于（并不一定是均匀的）所有多项式规模的电路族的保密性要求。假定非均匀单向函数（见 2.2 节中的定义 2.2.6）是存在的，使用和均匀情况下相同的构造方法，可以构造具有非均匀保密性要求的承诺方案。因此，我们有：

定理 4.4.6：假设存在非均匀单向函数（如定义 2.2.6 中定义的那样），那么存在一个比特承诺方案（如定义 4.4.1 中定义的那样），使得保密性条件对于多项式规模的电路也是成立的。

4.4.2　图着色的零知识证明

给出一种 ***NP*** 完备语言的零知识证明系统意味着对于 ***NP*** 中的每种语言都存在零知识证明系统。这个直观的吸引人的结论确实需要进行证明，我们将这个证明推迟到后面进行。在本节中，我们给出一种 ***NP*** 完备语言，图三着色的零知识证明系统。这种选择事实上是任意的。

语言图的三着色记为 $G3C$，包括所有满足下面条件的简单（有限）图（即，没有重边和自环的图）[①]：这些图可以使用 3 种颜色对顶点进行着色，使得任意两个相邻的顶点都具有不同的颜色。正式地说，如果存在一个映射 $\phi: V \to \{1,2,3\}$，使得对于每个 $(u,v) \in E$，都有 $\phi(u) \neq \phi(v)$，那么图 $G=(V,E)$ 就是可三着色的图。

4.4.2.1　动机的讨论

$G3C$ 零知识证明系统的思想出发点是，将一个图是可三着色的这样的结论的证明分解为排列在模板上的多项式数量的块，这些模板的每一个都不能产生知识，但是将它们重叠在一起就可以证明结论的有效性。假定证明者产生了这样的信息片断，并将每个片断分别放置到不同的密封且不透明的信封中，并且允许验证者打开并检查组成其中一个模板的那些信息片断。那么当然验证者肯定不会在这个过程中获得任何知识，然而它对结论的有效性（这个图是可三着色的）的信心增加了。这种抽象想法的具体实现如下。

为了证明图 $G=(V,E)$ 是可三着色的，证明者产生这个图的一个随机三着色，记为 ϕ（实际上是对一个固定着色的随机标号）。每个单个顶点的颜色组成了有关三着色的一条信息。模板集对应于边集（即，每对 $(\phi(u),\phi(v))$ 组成了一个模板，对应于结论 G 是可三着色的，其中 $(u,v)\in E$）。每个单独的模板（仅仅是一个随机取自 $\{1,2,3\}$ 的不同元素组成的对）都不会产

---

① 一个简单有限图是一个 (V,E) 对，其中 V 是一个有限集合，而 E 是 V 的 2 元素子集的集合；也就是说，$E\subseteq\{e\subseteq V: |e|=2\}$。$V$ 的元素叫做顶点，E 的元素叫做边。尽管每条边是一个 V 中两个元素的无序对，我们仍然使用有序对符号 $(u,v)\in E$，而不用符号 $\{u,v\}\in E$ 来表达。对于 $e=(u,v)\in E$，我们说 u 和 v 是 e 的端点或者说 u 与 v 是相邻的。

生知识。然而，如果所有的模板都是 OK 的(即，每个模板都包含一对取自{1,2,3}的不同元素)，那么图就一定是可三着色的。因此，不能三着色的图至少包含一个坏的模板，因此将以很大的概率被拒绝。下面是得到的 G3C 零知识交互证明系统的抽象描述。

- 公共输入：简单图 $G=(V,E)$。
- 证明者第一步：令 ψ 是 G 的一个三着色。证明者选择在{1,2,3}上的一个随机置换 π，并且对每个 $v\in V$，令 $\phi(v)\stackrel{\text{def}}{=}\pi(\psi(v))$。这样，证明者就形成了一个对于三着色 ψ 的随机重标记。证明者发送给验证者 $|V|$ 个锁着的不透明的盒子，其中第 v 个盒子中包含数值 $\phi(v)$。
- 验证者第一步：验证者均匀选择一条边 $(u,v)\in E$，并将它发送给证明者。
- 动机说明：验证者要求检验顶点 u 和 v 的颜色。
- 证明者第二步：证明者将对应盒子 u 和 v 钥匙发送给验证者。
- 验证者第二步：验证者打开盒子 u 和 v，并且当且仅当它们包含两个取自{1,2,3}的不同元素时接受。

显然，如果输入图是可三着色的，那么证明者就可以使验证者总是接受。反过来，如果输入图不是可三着色的，那么放置在盒子中的任何内容都必然至少在一条边上不满足条件，因此验证者将会以至少 $1/|E|$ 的概率拒绝。因此，前述的协议在输入是 G3C 图和输入不是 G3C 图的情况下的接受概率之间具有很大差距。零知识特性很容易从这样的抽象环境中得出，因为可以通过在由验证者指定的盒子中放置一个不同颜色的随机对来模拟真正的交互。我们强调，这种简单的讨论在数字实现时是不可能的，因为盒子并不是完全不受它的内容影响(而是相当受影响，只不过是以一种不能辨别的方式进行影响)。最后，我们指出对于结论(输入图是可三着色的)有效性的信任度可以通过顺序地应用前面的证明足够多次来提高(实际上，如果盒子如假设的一样是完备的，那么也可以使用并行重复上面的证明来提高对结论有效性的信任度；然而，盒子在下面提出的数字实现中不是完备的)。

4.4.2.2 交互证明

现在我们讨论抽象协议的数字实现。在这个实现中，盒子通过承诺方案来实现。也就是说，对于每个盒子，我们都会调用承诺方案的一个独立的执行。这样会使我们仅为某些承诺执行公布阶段，这个特性对我们的方案至关重要。为了描述简单，我们使用构造 4.4.2 中提出的简单承诺方案(或者，更通用一些，使用任何单向交互承诺方案)。我们用 $C_s(\sigma)$ 表示发送者使用硬币 s，对(三重)数值 σ 的承诺。

构造 4.4.7(图三着色零知识证明)：

- 公共输入：一个简单(可三着色的)图 $G=(V,E)$。令 $n\stackrel{\text{def}}{=}|V|$，其中 $V=\{1,\ldots,n\}$。
- 证明者的辅助输入：G 的一个三着色 ψ。
- 证明者的第一步(P1)：证明者选择一个在{1,2,3}上的随机置换 π，对每个 $v\in V$，令 $\phi(v)\stackrel{\text{def}}{=}\pi(\psi(v))$。证明者使用承诺方案来将它自己承诺为每个顶点的颜色。也就是说，证明者均匀独立地选择 $s_1,\cdots,s_n\in\{0,1\}^n$，对每个 $i\in V$ 计算 $c_i=C_{s_i}(\phi(i))$，并将 $c_1,\ldots,c_n$ 发送给验证者。
- 验证者的第一步(V1)：验证者均匀地选择一条边 $(u,v)\in E$，并将它发送给证明者。
- 证明者的第二步(P2)：不失一般性，我们可以假设从验证者那里收到的消息是一条边，

记为(u,v)（否则，证明者将(u,v)设置为G预先确定的某条边）。证明者使用承诺方案的（规范）公布阶段来将顶点u和v的颜色公布给验证者。即，证明者将$(s_u,\phi(u))$和$(s_v,\phi(v))$发送给验证者。

- 验证者的第二步（$V2$）：验证者检验与承诺u和v相应的数值是否被正确地公布，并检验这些数值是否是不同的。即，当接收到(s,σ)和(s',τ)时，验证者检验$c_u=C_s(\sigma)$，$c_v=C_{s'}(\tau)$和$\sigma\neq\tau$（σ和τ都是属于$\{1,2,3\}$的）是否成立。如果所有条件都成立，验证者接受；否则，拒绝。

我们记这个证明者程序为P_{G3C}。

我们强调，验证者程序和证明者程序可以在概率多项式时间内实现。对于证明者程序，这个特性通过使用证明者的辅助输入来实现。如同我们将要看到的一样，前述的协议构成了一个$G3C$的弱交互证明。通常，可信度可以通过重复足够多次连续应用来增加（即，错误概率可以降低）。然而，$G3C$交互证明的存在是显然的（因为$G3C\in$ **NP**）。妙处在于这个协议是（关于辅助输入）零知识的。使用顺序合成引理（引理4.3.11），可以得到重复这个协议多项式次数的连续应用将保持零知识特性。

命题4.4.8：假设构造4.4.7中使用的承诺方案满足（非均匀）保密性要求和明确性要求，那么构造4.4.7构成一个$G3C$的辅助输入零知识（广义）交互证明。

关于构造4.4.7的进一步讨论，见4.4.2.4节。

4.4.2.3 模拟器：命题4.4.8的证明

我们首先证明构造4.4.7构成一个$G3C$弱交互证明。首先假定输入图确实是可三着色的，那么如果证明者遵循指定的程序，验证者就总会接受（即，以概率1接受）。另一方面，如果输入图不是可三着色的，那么不管证明者怎样做，在步骤P1发送的n个承诺都不能对应于图的一个三着色（因为不存在这样的着色）。我们强调，承诺与数值的惟一对应通过承诺方案的明确性保证。因此，一定存在一条边$(u,v)\in E$，使得在步骤P1中发送的c_v和c_v不是对$\{1,2,3\}$内两个不同元素的承诺。因此，不管证明者怎样做，验证者都会以至小$1/|E|$的概率拒绝。因此，在输入在$G3C$中和输入不在$G3C$中这两种情况下的接受概率之间就存在一个较大的（以输入长度来衡量的）差距。

现在我们将说明在构造4.4.7中指定的证明者程序P_{G3C}事实上是$G3C$零知识的。这个结论不参照（验证者的）辅助输入就能证明，但是将这个结论扩展为辅助输入零知识可以直接得到。同样，我们使用零知识的可选定义（即定义4.3.3），并且显示怎样模拟每个概率多项式时间交互机V^*与P_{G3C}交互的观点。正如同构图证明系统（即构造4.3.8）一样，如果验证者遵循指定的程序，那么模拟验证者对与P_{G3C}交互的观点是简单的。然而，我们需要模拟一般情况下的验证者观点（在这种情况下，验证者使用任意多项式时间交互程序）。下面是我们模拟（即对于任意V^*，我们关于模拟器M^*的结构）的一个概述。

模拟器M^*集成了交互程序V^*的代码。对于输入图$G=(V,E)$，模拟器M^*（没有使用G的三着色）首先均匀独立地选择n个值$e_1,\ldots,e_n\in\{1,2,3\}$，并且对这些值中的每一个都构造承诺（这些$e_i$构成了图的一个"伪着色"，在这个"伪着色"中每条边的端点都会以$\frac{2}{3}$的概率着不同的颜色）。在这个过程中，尽管模拟器与P_{G3C}的行为很不相同，但是这样产生的承诺序列与步骤P1中P_{G3C}发送的对有效三着色的承诺序列是计算不可分辨的。当给定模拟器产生的承

诺时，如果 V^* 要求检查一条边 (u,v) 并满足 $e_u \neq e_v$，那么模拟器事实上可以正确回答，并且这样做的时候它完成了对与 P_{G3C} 交互时验证者观点的模拟。然而，如果 V^* 要求检查一条边 (u,v) 并满足 $e_v = e_u$，那么模拟器就无法正确回答，我们输出 $\perp$ 并终止模拟器的运行。我们强调，我们并没有假设模拟器预先就“知道”验证者将会要求检查哪一条边。模拟器的有效性源自一个不同的信息源。如果验证者的请求忘记了证明者的承诺，那么验证者将以 $\frac{2}{3}$ 的概率要求检查一条已经恰当着色的边。使用承诺方案的保密性，可以得到验证者的请求都“几乎忘却”了承诺的数值。零知识的结论就得到了（但是，付出了一定的努力）。进一步的细节在后面讲解。我们从对模拟器的详细描述开始。

模拟器 M^* 对于一个输入图 $G=(V,E)$，其中 $n=|V|$，模拟器 M^* 按如下步骤运行：

1. 设置 V^* 的随机磁带：令 $q(\cdot)$ 为限定 V^* 的运行时间的多项式。模拟器 M^* 以均匀选择一个字符串 $r \in \{0,1\}^{q(n)}$ 来开始运行，这个字符串将作为 V^* 本地随机磁带的内容。

2. 模拟证明者的第一步(P1)：模拟器 M^* 均匀独立地选择 n 个值 $e_1,\ldots,e_n \in \{1,2,3\}$ 和 n 个用于承诺这些值的随机字符串 $s_1,\ldots,s_n \in \{0,1\}^n$。对每个 $i \in V$，模拟器计算承诺 $d_i = C_{s_i}(e_i)$。

3. 模拟验证者的第一步(V1)：模拟器 M^* 通过将 G 放置在 V^* 的公共输入磁带上，将 r（在步骤 1 中选择的）放置在 V^* 的本地随机磁带上，将序列 $(d_1,\ldots,d_n)$（在步骤 2 中构造的）放置在 V^* 的输入消息磁带上启动 V^* 的执行。在执行 V^* 的步骤多项式次后，模拟器就可以阅读 V^* 的输出消息了，这个消息记为 m。不失一般性，我们假设 $m \in E$，并令 $(u,v)=m$（实际上，对 $m \notin E$ 的边同 P_{G3C} 的步骤 P2 中的边一样处理，即 (u,v) 被设定为 G 中预先指定的边）。

4. 模拟证明者的第二步(P2)：如果 $e_u \neq e_v$，那么模拟器就会输出 $(G,r,(d_1,\ldots,d_n),(s_u,e_u,s_v,e_v))$ 并停止运行。

5. 模拟失败：否则（即 $e_u = e_v$），模拟器就会输出 $\perp$ 并停止运行。

因为假定 V^* 是多项式时间的，那么 M^* 也应该是多项式时间的。剩下要证明的是 M^* 至多以 $\frac{1}{2}$ 的概率输出 $\perp$，如果不输出 $\perp$，模拟器的输出就与验证者对“与 P_{G3C} 的真实交互”的观点是计算不可分辨的。这个命题通过运行模拟器 n 次，并且第一次不输出 $\perp$ 就可以得到证明。我们现在先证明两个结论。

结论 4.4.8.1：对每个足够大的图 $G=(V,E)$，$M^*(G)=\perp$ 的概率上界为 $\frac{1}{2}$。

（实际上，可以证明一个更强的结论：对于每个多项式 p 和所有足够大的图 $G=(V,E)$，$M^*(G)=\perp$ 的概率上界为 $\frac{1}{3}+\frac{1}{p(|V|)}$）

证明：我们用 $p_{u,v}(G,r,(e_1,\ldots,e_n))$ 来表示对所有的 $s_1,\ldots,s_n \in \{0,1\}^n$，当输入图为 G，随机硬币为 r，证明者消息为 $(C_{s_1}(e_1),\ldots,C_{s_n}(e_n))$ 时，V 用消息 (u,v) 来回答的概率。为了简单起见，我们假设 V^* 总是用 G 的边来回答（因为否则的话，它的消息被当作是 G 的一条边来对待）。我们首先指出，对于每个足够大的图 $G=(V,E)$，每个 $r \in \{0,1\}^{q(n)}$，每条边 $(u,v) \in E$ 和每两个序列 $\alpha,\beta \in \{1,2,3\}^n$，都有：

$$|p_{u,v}(G,r,\alpha) - p_{u,v}(G,r,\beta)| \leq \frac{1}{2|E|} \tag{4.2}$$

实际上，我们可以证明如下子结论。

请求遗忘子结论：对于每个多项式 $p(\cdot)$，每个足够大的图 $G=(V,E)$，每个 $r \in \{0,1\}^{q(n)}$，

每条边$(u,v)\in E$ 和每两个序列 $\alpha,\beta\in\{1,2,3\}^n$，都有：

$$|p_{u,v}(G,r,\alpha)-p_{u,v}(G,r,\beta)|\leq\frac{1}{p(n)}$$

请求遗忘子结论可以使用承诺方案的非均匀秘密性来证明。读者应该能够将这个证明的细节填充完整(尽管如此，我们还是给出这个子结论的证明)。

请求遗忘子结论的证明：反证法。假设存在一个多项式 $p(\cdot)$和一个无限整数序列，满足对于(序列中的)每个整数 n，都存在一个具有 n 个顶点的图 $G_n=(V_n,E_n)$，一个字符串 $r_n\in\{0,1\}^{q(n)}$，一条边$(u_n,v_n)\in E_n$，两个序列 $\alpha_n,\beta_n\in\{1,2,3\}^n$，使得

$$|p_{u_n,v_n}(G_n,r_n,\alpha_n)-p_{u_n,v_n}(G_n,r_n,\beta_n)|>\frac{1}{p(n)}$$

我们构造一个电路族$\{A_n\}$，其中 A_n 集成了交互机 V^*，图 G_n 和 $r_n,u_n,v_n,\alpha_n,\beta_n$ 都与矛盾假设中的一样。对于输入 y(是 α_n 或者 β_n 的承诺序列)，电路 A_n 运行 V^*(对于输入 G_n，硬币 r_n 和证明者的消息 y)，当且仅当 V^* 回复(u_n,v_n)时输出 1。显然，$\{A_n\}$是一个(非均匀)多项式规模电路族。关键的结论是 A_n 能分辨 α_n 的承诺和 β_n 的承诺，这是因为：

$$\Pr[A_n(C_{U_n^{(1)}}(e_1),\ldots,C_{U_n^{(n)}}(e_n))=1]=p_{u_n,v_n}(G_n,r_n,(e_1,\ldots,e_n))$$

其中 $U_n^{(i)}$ 为在$\{0,1\}^n$ 上服从均匀分布的独立随机变量。通过标准的混合理论(混合理论将承诺序列的不可分辨性与单个承诺的不可分辨性关联起来)得到与承诺方案的(非均匀)保密性相矛盾的结论。

回到结论 4.4.8.1 的证明上来，我们现在运用这个子论题来限定模拟器输出上的概率上界。直觉上很简单：因为 V^* 的请求几乎忘记模拟器用来承诺它自己的数值，V^* 请求检验不合法着色的边比不看承诺就请求检验不合法着色的边更频繁是不可能的。因此，V^* 以大约$\frac{1}{3}$的概率来要求检验不合法着色的边，因此 $\Pr[M^*(G)=\perp]\approx\frac{1}{3}$。更为严格(但简单明了)的分析如下。

令 $M_r^*(G)$表示在输入为 G，步骤 1 中 M^* 选择字符串 r 的条件下，机器 M^* 的输出。我们要提醒读者，只有当验证者对于输入 G、随机磁带 r 和对某个伪着色$(e_1,\ldots,e_n)$的承诺，要求检验一条非法着色的边(u,v)(即 $e_u=e_v$)时，才会有 $M_r^*(G)=\perp$。令 $E_{(e_1,\ldots,e_n)}$ 表示关于$(e_1,\ldots,e_n)$，非法着色(即满足 $e_u=e_v$)的边$(u,v)\in E$ 的集合。那么，任意选定一个 r，并考虑 $\bar{e}=(e_1,\ldots,e_n)\in\{1,2,3\}^n$ 的所有可能的选择，我们有

$$\Pr[M_r^*(G)=\perp]=\sum_{\bar{e}\in\{1,2,3\}^n}\frac{1}{3^n}\cdot\sum_{(u,v)\in E_{\bar{e}}}p_{u,v}(G,r,\bar{e})$$

(注意，$p_{u,v}(G,r,\bar{e})$表示当给定对数值 $\bar{e}$ 的随机承诺序列时验证者要求检验(u,v)的概率。)定义 $B_{u,v}$ 为满足 $e_u=e_v$ 的 n 元组$(e_1,\ldots,e_n)\in\{1,2,3\}^n$ 的集合。显然，$|B_{u,v}|=3^{n-1}$，并且

$$\begin{aligned}\{(\bar{e},(u,v)):\bar{e}\in\{1,2,3\}^n\&(u,v)\in E_{\bar{e}}\}&=\{(\bar{e},(u,v)):\bar{e}\in\{1,2,3\}^n\&e_u=e_v\}\\&=\{(\bar{e},(u,v)):(u,v)\in E\&\bar{e}\in B_{u,v}\}\end{aligned}$$

通过直接计算，我们得到

$$\begin{aligned}\Pr[M_r^*(G)=\perp]&=\frac{1}{3^n}\cdot\sum_{\bar{e}\in\{1,2,3\}^n}\sum_{(u,v)\in E_{\bar{e}}}p_{u,v}(G,r,\bar{e})\\&=\frac{1}{3^n}\cdot\sum_{(u,v)\in E}\sum_{\bar{e}\in B_{u,v}}p_{u,v}(G,r,\bar{e})\end{aligned}$$

$$\leq \frac{1}{3^n} \cdot \sum_{(u,v)\in E} |B_{u,v}| \cdot \left(p_{u,v}(G,r,(1,\ldots,1)) + \frac{1}{2|E|}\right)$$
$$= \frac{1}{6} + \frac{1}{3} \cdot \sum_{(u,v)\in E} p_{u,v}(G,r,(1,\ldots,1))$$
$$= \frac{1}{6} + \frac{1}{3}$$

其中等号成立的条件是 4.2 式取等号。这样就证明了结论。

为了简单起见，在接下来的讨论中我们假设，对于公共输入 $G\in G3C$，证明者将把按照字典顺序排列的第一个 G 的三着色作为辅助输入。这使得我们在表示符号中可以忽略 P_{G3C} 的辅助输入（它被暗含在公共输入中）。这种讨论可以很容易地扩展到一般情况，在一般情况下，P_{G3C} 以 G 的任意一个三着色作为辅助输入。

结论 4.4.8.2：对于输入 $G=(V,E)\in G3C$，由 M^* 的不为 $\perp$ 的输出构成的总体与总体 $\{\text{view}_{V^*}^{P_{G3C}}(G)\}_{G\in G3C}$ 是计算不可分辨的。也就是说，对于每个概率多项式时间算法 A，每个多项式 $p(\cdot)$ 和所有足够大的图 $G=(V,E)$，都有

$$\left|\Pr[A(M^*(G))=1 \mid M^*(G)\neq\perp] - \Pr\left[A(\text{view}_{V^*}^{P_{G3C}}(G))=1\right]\right| < \frac{1}{p(|V|)}$$

我们强调，这些总体是很不相同的（即，它们之间的统计距离非常接近于最大可能距离），然而它们仍然是计算不可分辨的。实际上，我们也可以通过多项式规模的（非均匀）电路族来证明这些总体是不可分的。乍看起来，结论 4.4.8.2 可以简单地从承诺方案的保密性得到。事实上，结论 4.4.8.2 可以使用承诺方案的保密性进行证明，但是证明比我们最初想象的要复杂得多。困难在于，前述的总体不仅包括对数值的承诺，而且还包括对某些值的公开。而且，选择公开哪个承诺依赖于整个承诺序列（我们利用了公开的承诺数目是恒定的这个事实）。

证明：令 $m^*(G)$ 表示 $M^*(G)$ 在 $M^*(G)\neq\perp$ 时的分布。对于任何算法 A，我们用 $\varepsilon_A(G)$ 来表示 A 关于这个结论中总体的分辨距离；即，

$$\varepsilon_A(G) \stackrel{\text{def}}{=} \left|\Pr[A(m^*(G))=1] - \Pr\left[A(\text{view}_{V^*}^{P_{G3C}}(G))=1\right]\right| \tag{4.3}$$

我们的目的是要证明对于每个概率多项式时间算法 A，$\varepsilon_A(G)$ 的数值作为 G 的顶点数目的函数是可以忽略的。回想对于 $G=(V,E)$，$m^*(G)$ 和 $\text{view}_{V^*}^{P_{G3C}}(G)$ 都是形如 $(r,\{\alpha_1,\ldots,\alpha_{|V|}\},(u,v),(s_u,\sigma_u,s_v,\sigma_v))$ 的序列，其中 $r\in\{0,1\}^{q(|V|)}$，$(u,v)\in E$，$\sigma_u\neq\sigma_v\in\{1,2,3\}$，$\alpha_u=C_{s_u}(\sigma_u)$，并且 $\alpha_v=C_{s_v}(\sigma_v)$。在两种情况下，$(u,v)$ 被称为验证者请求。

给定一个图 $G=(V,E)$，我们为每条边 $(u,v)\in E$ 定义两个随机变量，分别描述在验证者的请求等于 (u,v) 时 M^* 的输出和 V^* 对真实交互的观点。特别地：

- $\mu_{u,v}(G)$ 用来描述验证者的请求等于 (u,v) 条件下的 $M^*(G)$（等同于 $m^*(G)$）。
- $v_{u,v}(G)$ 用来描述验证者的请求等于 (u,v) 条件下的 $\text{view}_{V^*}^{P_{G3C}}(G)$。

令 $p_{u,v}(G)$ 表示 $m^*(G)$ 使得验证者的请求等于 (u,v) 的概率。相似地，令 $q_{u,v}(G)$ 为 $\text{view}_{V^*}^{P_{G3C}}(G)$ 使验证者的请求等于 (u,v) 的概率。

与结论相反，假定在结论中提到的总体是计算可分辨的，那么下面两种情况至少有一种会发生。

情况 1：在与 P_{G3C} 进行交互时 V^* 请求的概率轮廓与验证者对于用 $m^*(G)$ 表示的输出中

请求的概率轮廓之间存在不可忽略的差别。正式地说，存在一个多项式 $p(\cdot)$ 和一个无限整数序列，使得对于(序列中)每个整数 n，都存在一个 n 节点的图 $G_n=(V_n,E_n)$ 和一条边 $(u_n,v_n)\in E_n$，满足

$$|p_{u_n,v_n}(G_n)-q_{u_n,v_n}(G_n)|>\frac{1}{p(n)}$$

否则，对于每个多项式 p'，除去有限多个 G，以及图 $G=(V,E)$ 中的所有边 (u,v)，都有：

$$|p_{u,v}(G)-q_{u,v}(G)|\leq\frac{1}{p'(|V|)} \tag{4.4}$$

情况 2：分辨前述总体的算法也能分辨在 V^* 作特殊请求的情况下的总体。而且，这个请求发生的不可忽略的概率对这两种总体是大概相同的。正式地说，存在一个概率多项式时间算法 A，一个多项式 $p(\cdot)$ 和一个无限整数序列，使得对于(序列中的)每个整数 n，都存在一个 n 节点的图 $G_n=(V_n,E_n)$ 和一条边 $(u_n,v_n)\in E_n$，满足下面条件：

- $q_{u_n,v_n}(G_n)>\dfrac{1}{p(n)}$
- $|p_{u_n,v_n}(G_n)-q_{u_n,v_n}(G_n)|<\dfrac{1}{3\cdot p(n)^2}$
- $\left|\Pr[A(\mu_{u_n,v_n}(G_n))=1]-\Pr[A(v_{u_n,v_n}(G_n))=1]\right|>\dfrac{1}{p(n)}$

如果情况 1 不满足，那么情况 2 就一定满足，这个事实可以通过对概率空间按照公布的边进行分解来得到。如下是证明的细节。

考虑算法 A，这个算法将模拟器的输出与对无限多个图 $G=(V,E)$ 的真实交互分辨开来，其中分辨距离为 G 的测度的多项式的倒数；即 $\varepsilon_A(G)>1/\mathrm{poly}(|V|)$。令 $\mathrm{req}_{u,v}(\alpha)$ 表示事件：副本为 α，验证者的请求等于 (u,v)。那么在 G 中就必然存在一条边 (u,v)，使得

$$\begin{aligned}&|\Pr[A(m^*(G))=1\,\&\,\mathrm{req}_{u,v}(m^*(G))]\\&-\Pr\left[A(\mathrm{view}^{P_{G3C}}_{V^*}(G))=1\,\&\,\mathrm{req}_{u,v}(\mathrm{view}^{P_{G3C}}_{V^*}(G))\right]\Big|\geq\frac{\varepsilon_A(G)}{|E|}\end{aligned}$$

注意到，

$$p_{u,v}(G)=\Pr[\mathrm{req}_{u,v}(m^*(G))]$$

$$q_{u,v}(G)=\Pr\left[\mathrm{req}_{u,v}(\mathrm{view}^{P_{G3C}}_{V^*}(G))\right]$$

$$\Pr[A(\mu_{u,v}(G))=1]=\Pr[A(m^*(G))=1\mid\mathrm{req}_{u,v}(m^*(G))]$$

$$\Pr[A(v_{u,v}(G))=1]=\Pr\left[A(\mathrm{view}^{P_{G3C}}_{V^*}(G))=1\,\middle|\,\mathrm{req}_{u,v}(\mathrm{view}^{P_{G3C}}_{V^*}(G))\right]$$

因此，在其中一些符号中忽略 G，就可以得到

$$|p_{u,v}\cdot\Pr[A(\mu_{u,v}(G))=1]-q_{u,v}\cdot\Pr[A(v_{u,v}(G))=1]|\geq\frac{\varepsilon_A(G)}{|E|}$$

令 $p(|V|)\overset{\text{def}}{=}\frac{2|E|}{\varepsilon_A(G)}$（即 $\frac{\varepsilon_A(G)}{|E|}=\frac{2}{p(|V|)}$），并运用 4.4 式（及 $p'=3p^2$），我们可以得到 $|p_{u,v}-q_{u,v}|<\frac{1}{3p(|V|)^2}$ 和

$$|q_{u,v}\cdot\Pr[A(\mu_{u,v}((G))=1]-q_{u,v}\cdot\Pr[A(v_{u,v}(G))=1]|>\frac{1}{p(|V|)}$$

对于所有的有限个图 G 都成立。

因此，$q_{u,v}>1/p(|V|)$ 和

$|\Pr[A(\mu_{u,v}(G))=1]-\Pr[A(v_{u,v}(G))-1]|>1/p(|V|)$

也将成立。

情况 1 可以立即排除，因为情况 1 很容易导致矛盾（与承诺方案的非均匀保密性矛盾）：想法就是使用结论 4.4.8.1 的证明中出现的请求遗忘子结论。我们忽略了细节。因此我们只要讨论情况 2 即可。

现在我们说明情况 2 也将会导致矛盾。为了说明问题，我们将构造一个电路族，这个电路族可以将对不同的数值序列的承诺区别开。有趣的是，这些序列中任意一个序列都不等于证明者或模拟器产生的承诺序列。下面是构造的概况。第 n 个电路得到一个由 $3n$ 个承诺构成的序列，并由这个序列产生一个由 n 个承诺组成的序列（这个序列的其中一部分是输入的子序列）。当电路的输入序列来自同一个分布时，电路根据证明者产生的承诺序列来产生一个子序列。同样，当（电路的）输入序列来自其他分布时，电路将会根据模拟器产生的序列来产生子序列。我们强调，电路这样动作是在不知道输入究竟选择的是哪个分布的情况下进行的。在产生一个 n 位长的序列之后，电路将这个序列提供给 V^*，并且根据 V^* 的行为，电路会将序列的一部分提供给算法 A（在情况 2 下提到的算法）。下面是有关电路族的细节描述。

令 ψ_n 为证明者运用的 $G_n=(V_n,E_n)$ 的（按字典顺序第一个）三着色，其中 $V_n=\{1,...,n\}$。我们构造一个电路族 $\{A_n\}$，令 A_n 集成了交互机 V^*、"分辨"算法 A、图 G_n、三着色 ψ_n 和边 (u_n,v_n)，所有这些与情况 2 中的都相同。电路 A_n 的输入将会是一个由 $3n$ 个数值承诺组成的序列，每个承诺都取值于 $\{1,2,3\}$。电路可以分辨对均匀选择的 $3n$ 位长序列的承诺和对固定序列 $1^n2^n3^n$（也就是，包含 n 个连续的 1、n 个连续的 2 和 n 个连续的 3 的序列）的承诺。下面是 $\mathbf{A}_n$ 的操作的描述。在这个描述中，对于 $e\in\{1,2,3\}$，我们用 $C(e)$ 来表示通过均匀选择 $s\in\{0,1\}^n$ 并输出 $C_s(e)$ 得到的随机变量。我们将这个概念扩展到 $\{1,2,3\}$ 上的序列（即，$C(e_1,...,e_t)=C(e_1),...,C(e_t)$，其中独立的随机化过程在每个承诺中都使用）。

$\mathbf{A}_n$ 的操作：对于输入 $y=(y_1,...,y_{3n})$（其中每个 y_i 都被认为是一个对 $\{1,2,3\}$ 中元素的承诺），电路 A_n 如下执行：

- A_n 首先均匀地选择一个 $\{1,2,3\}$ 上的置换 π，并为每个 $i\in V_n$ 计算 $\phi(i)=\pi(\psi_n(i))$。

值得注意的是，$(\phi(u_n),\phi(v_n))$ 均匀地分布在 6 个可能的 $\{1,2,3\}$ 的不同元素对中。

- 对于每个 $i\in V_n\backslash\{u_n,v_n\}$，电路令 $c_i=y_{\phi(i)\cdot n-n+i}$（即，如果 $\phi(i)=1$ 则 $c_i=y_i$，如果 $\phi(i)=2$ 则 $c_i=y_{n+i}$，如果 $\phi_{(i)}=3$ 则 $c_i=y_{2n+i}$）

注意，每个 y_j 最多只用一次，这些 y_j 中有 $2n+2$ 个根本没有用到过。

- 电路均匀地选择 $s_{u_n},s_{v_n}\in\{0,1\}^n$，并令 $c_{u_n}=C_{s_{u_n}}(\phi(v_n))$ 和 $c_{v_n}=C_{s_{v_n}}(\phi(v_n))$ 成立。

在 y 取自分布 $C(1^n2^n3^n)$ 的情况下，刚建立的序列 $c_1,...,c_n$ 与步骤 1 中证明者发送的承诺序列是同分布的。另一方面，假定 y 在所有 $3n$ 位长序列的所有可能承诺上服从均匀分布（即，y 是通过均匀地选择 $\alpha\in\{1,2,3\}^{3n}$ 并输出 $C(\alpha)$ 来构造的）。那么在端点 u_n 和 v_n 被分配不同颜色的情况下，刚才建立的序列 $c_1,...,c_n$ 与步骤 2 中由模拟器形成的承诺序列同分布。

- 电路通过将 G_n 放置在 V^* 的公共输入磁带上，将均匀选择的 $r\in\{0,1\}^{q(n)}$ 放置在 V^* 的本地随机磁带上，将序列 $(c_1,...,c_n)$ 放置在 V^* 的输入消息磁带上来启动 V^* 的执行。电路读取 V 的输出消息，记作 m。

- 如果 $m \neq (u_n, v_n)$，那么电路输出 0。
- 否则（即 $m = (u_n, v_n)$），电路将会调用算法 A，并输出

$$A(G_n, r, (c_1, \ldots, c_n), (s_{u_n}, \phi(u_n), s_{v_n}, \phi(v_n)))$$

显然，A_n的大小是 n 的多项式。我们现在估计 A_n 的分辨能力。我们首先考虑在输入一个对序列 $1^n2^n3^n$ 的随机承诺时，电路 A_n 输出 1 的概率。读者可以很容易地验证，由电路 A_n 构造的序列 $(c_1, \ldots, c_n)$ 与步骤 P1 中证明者发出的序列是同分布的。因此，采用前面引入的符号，我们可以得到

$$\Pr[A_n(C(1^n2^n3^n)) = 1] = q_{u_n,v_n}(G_n) \cdot \Pr[A(v_{u_n,v_n}(G_n)) = 1]$$

另一方面，在输入一个对在 $\{1,2,3\}$ 上均匀选择的 $3n$ 位长序列的随机承诺时，我们考虑电路 A_n 输出 1 的概率。读者可以很容易地验证，在 $e_{u_n} \neq e_{v_n}$ 成立时，由电路 A_n 构造的序列 $(c_1, \ldots, c_n)$ 与在步骤 2 中由模拟器产生的序列 $(d_1, \ldots, d_n)$ 是同分布的（其中 $d_i = C(e_i)$）。令 T_{3n} 为在 $\{1,2,3\}^{3n}$ 上服从均匀分布的随机变量，我们可以得到

$$\Pr[A_n(C(T_{3n})) = 1] = p'_{u_n,v_n}(G_n) \cdot \Pr[A(\mu_{u_n,v_n}(G_n)) = 1]$$

其中 $p'_{u_n,v_n}(G_n)$ 表示在 $e_{u_n} \neq e_{v_n}$ 成立时，在模拟的步骤 3 中验证者以 (u_n, v_n) 做答的概率。因为结论 4.4.8.1 的证明实际上确定了，$\left|\Pr[M^*(G_n) \neq \bot] - \frac{2}{3}\right|$ 是关于 n 可忽略的，所以 $|p'_{u_n,v_n}(G_n) - p_{u_n,v_n}(G_n)| < \frac{1}{3 \cdot p(n)^2}$ 对除了有限个之外的所有 G_n 都满足。

对于最后声明的证明： 注意到 $p'_{u_n,v_n}(G_n)$ 和 $p_{u_n,v_n}(G_n)$ 所指的是在不同条件空间（即，一个是 $e_{u_n} \neq e_{v_n}$，而另一个是 $M^*(G_n) \neq \bot$）下的相同事件（即 V^* 的请求等于 (u_n, v_n)）。实际上，在这两种情况下，考虑 V^* 的请求等于 (u_n, v_n) 且 $e_{u_n} \neq e_{v_n}$ 的事件都是很有益的。

记后一个事件为 X，我们有[①]

- $p'_{u_n,v_n}(G_n) = \Pr[X \mid e_{u_n} \neq e_{v_n}] = \Pr[X] / \Pr[e_{u_n} \neq e_{v_n}]$
- $p_{u_n,v_n}(G_n) = \Pr[X \mid M^*(G_n) \neq \bot] = \Pr[X] / \Pr[M^*(G_n) \neq \bot$

运用 $\Pr[e_{u_n} \neq e_{v_n}] = \dfrac{2}{3} \approx \Pr[M^*(G_n) \neq \bot]$，其中 $\approx$ 为约等于符号（对于 n），这就说明 $|p'_{u_n,v_n}(G_n) - p_{u_n,v_n}(G_n)|$ 是可忽略的（对于 n）。

由情况 2 的条件，并在符号中忽略 G_n，就有：

$$|p'_{u_n,v_n} - q_{u_n,v_n}| \leq |p'_{u_n,v_n} - p_{u_n,v_n}| + |p_{u_n,v_n} - q_{u_n,v_n}| < \frac{2}{3 \cdot p(n)^2}$$

结合前面，我们可以得到

$$\begin{aligned}&|\Pr[A_n(C(1^n2^n3^n)) = 1] - \Pr[A_n(C(T_{3n})) = 1]| \\ &= |q_{u_n,v_n} \cdot \Pr[A(v_{u_n,v_n}) = 1] - p'_{u_n,v_n} \cdot \Pr[A(\mu_{u_n,v_n}) = 1]|\end{aligned}$$

---

① 为了进一步证明如下等式，令 X' 为 V^* 的请求等于 (u_n, v_n) 这一事件，注意到 X 为 X' 和 $e_{u_n} \neq e_{v_n}$ 的逻辑乘积。那么：

- 由定义，$p'_{u_n,v_n}(G_n) = \Pr[X' \mid e_{u_n} \neq e_{v_n}]$，而 $\Pr[X' \& e_{u_n} \neq e_{v_n}] / \Pr[e_{u_n} \neq e_{v_n}] = \Pr[X] / \Pr[e_{u_n} \neq e_{v_n}]$。
- 由定义 $p_{u_n,v_n}(G_n) = \Pr[X' \mid M^*(G_n) \neq \bot]$。注意到，$M^*(G_n) \neq \bot$ 和 X' 的逻辑乘积隐含着 $e_{u_n} \neq e_{v_n}$，因此前面的逻辑乘积就隐含了 X。另一方面，X 暗指 $M^*(G_n) \neq \bot$。因此 $\Pr[M^*(G_n) \neq \bot] \cdot \Pr[X' \mid M^*(G_n) \neq \bot] = \Pr[X' \& M^*(G_n) \neq \bot] = \Pr[X \& M^*(G_n) \neq \bot] = \Pr[X]$。我们可以得出结论 $p_{u_n,v_n}(G_n) = \Pr[X] / \Pr[M^*(G_n) \neq \bot]$。

$$\geq q_{u_n,v_n} \cdot |\Pr[A(v_{u_n,v_n}) = 1]1 - \Pr[A(\mu_{u_n,v_n} = 1]| - |p'_{u_n,v_n} - q_{u_n,v_n}|$$

$$> \frac{1}{p(n)} \cdot \frac{1}{p(n)} - \frac{2}{3 \cdot p(n)^2} = \frac{1}{3 \cdot p(n)^2}$$

因此,电路族$\{A_n\}$可以分辨对$\{1^n 2^n 3^n\}$的承诺和对$\{T_{3n}\}$的承诺。将平均理论和混合理论相结合,我们就可以得到这样的结论:存在一个可以分辨承诺的多项式规模的电路族。这与承诺方案的非均匀保密性相矛盾。□

现在已经得到了两种情况下的矛盾,结论 4.4.8.2 成立。■

将结论 4.4.8.1 与 4.4.8.2 相结合(并运用习题 9),P_{G3C} 的零知识特性就成立了。这样就完成了命题的证明。

4.4.2.4　得到的注释

构造 4.4.7 已经用单向承诺方案给出。这种方案的基本特性是,它们的保密性在很多(多项式个数)实例并发调用的时候也能够保持。命题 4.4.8 的证明实际上利用了这种特性。我们要指出的是,构造 4.4.4 也具有这种并发保密特性(尽管它不是单向的),因此,如果所采用的承诺方案是构造 4.4.4 中采用的那个方案,那么命题 4.4.8 就可以证明(见习题 15)。我们回想构造 4.4.4 构成承诺方案,当且仅当这样的方案确实存在(因为构造 4.4.4 是基于单向函数的,并且单向函数的存在性隐含在承诺方案的存在性之中)。

命题 4.4.8 假设存在非均匀保密承诺方案。这个命题的证明通过结合在不满足安全假设的电路中零知识特性失败的实例,使用了非均匀保密性。我们强调,有效(均匀)机器不一定会产生"坏"实例序列。换句话说,零知识需求具有一些非均匀特性。零知识的均匀模拟只要求,找到验证者能够获得知识的实例是不可能的(而不是根本不存在这样的实例)。运用均匀安全的承诺方案,可以证明构造 4.4.7 是均匀零知识的。

由于提供的是一个中等接受间距(在语言内部和语言外部的输入之间),构造 4.4.7 本身几乎没有什么实用价值。然而,对于公共输入 $G=(V,E)$,重复 $k \cdot |E|$ 次协议(并令验证者只有在所有迭代协议中都接受时才会接受),将会以 e^{-k} 为界的错误概率产生一个 $G3C$ 交互证明,其中 $e \approx 2.718$ 是自然对数基。即,对于公共输入 $G \notin G3C$,验证者总会接受;而对于公共输入 $G \in G3C$,验证者会以 e^{-k} 为上界的概率接受(不管证明者怎样做)。我们强调,根据顺序合成引理(引理 4.3.11),如果这些迭代是按顺序执行的,那么得到的(强)交互证明也是零知识的。令 k 为 $|G|$ 的超对数函数(例如,$k=|G|$),得到的交互证明的错误概率是可忽略的。我们指出,通过并行执行 $k \cdot |E|$ 次迭代产生的交互证明不一定是零知识的,见 4.5.4 小节。

构造 4.4.7 的一个重要特性是,如果给定公共输入图的一个三着色作为辅助输入,那么指定的证明者(即 P_{G3C})可以在概率多项式时间内实现。正如我们将会看到的一样,这个特性对于构造 4.4.7 在密码协议设计中的应用是至关重要的。

如前所述,选择 $G3C$ 作为 ***NP*** 完备语言的"切入点"完全是任意的。使用 4.4.2.1 节(即动机的讨论)中的思想,可以很容易地为其他流行的 ***NP*** 完备语言设计相似的零知识证明。

4.4.3　普遍结论和一些应用

图三着色的零知识证明(例如,构造 4.4.7)在理论上和实践中都具有重要性,这种重要性

源于这样一个事实，它可以在零知识情况下证明任何具有简短并且可被有效验证的证明的陈述。更精确地说，一个特定 **NP** 完备语言的零知识证明系统（例如，构造 4.4.7）可以用来为 **NP** 中的每种语言都给出零知识证明系统。

在为 **NP** 中的每种语言给出零知识证明系统之前，让我们先回想一些有关 $L\in$ **NP** 的约定和事实。我们首先知道每种语言是通过具有如下特性的二元关系 R 来刻划的：

- 存在多项式 $p(\cdot)$，使得对每个 $(x,y)\in R$，都有 $|y|\leq p(|x|)$。
- 存在用来决定 R 中的成员资格的多项式时间算法。
- $L=\{x:\exists w\ s.t.\ (x,\omega)\in R\}$。

（这样的 w 称为对 $x\in L$ 的成员资格的证据。）

实际上，**NP** 中的每种语言都可以通过有限多个这样的关系来描述。然而对于每个 $L\in$ **NP**，我们只选定并考虑一个特征性关系，记为 R_L。因为 $G3C$ 是 **NP** 完备的，我们知道 L 对于 $G3C$ 是多项式时间可约的（即，Karp 可约的）。即，存在一个多项式时间可计算函数 f，使得当且仅当 $f(x)\in G3C$ 时 $x\in L$。最后，我们注意到，从 L 到 $G3C$ 的标准约简记为 f_L，具有如下附加特性：

- 存在一个多项式时间可计算函数，记为 g_L，使得对于每个 $(x,w)\in R_L$，都有 $g_L(x,w)$是 $f_L(x)$的三着色。

我们强调，这个附加特性是 Karp 可约的标准定义不需要的。但是，可以很容易地证明（见习题 16），标准约简 f_L（即，L 到 SAT 的一般简化、SAT 到 $3SAT$ 的标准简化和 $3SAT$ 到 $G3C$ 的标准简化的组合）确实具有这样的一个相应的 g_L。使用这些约定，我们已经准备好将 **NP** 零知识证明构造"约简"为 $G3C$ 的零知识证明系统。

构造 4.4.9（语言 $L\in$ NP 的零知识证明）：

- 公共输入：字符串 x（假设在 L 中）。
- 证明者的辅助输入：对于 $x\in L$ 的成员资格的证据 ω（即，满足 $(x,w)\in R_L$ 的字符串 w）。
- 本地预计算：每一方都计算 $G\stackrel{\text{def}}{=}f_L(x)$。证明者计算 $\Psi\stackrel{\text{def}}{=}g_L(x,w)$。
- 调用 $G3C$ 零知识证明：各方都调用公共输入为 G 的零知识证明。证明者调用这个证明时还具有辅助输入 ψ。

显然，当给定一个 **NP** 证据（即，一个三着色）作为辅助输入时，如果在 $G3C$ 证明系统中指定的证明者可以在概率多项式时间内实现，那么构造 4.4.9 中的证明者也能够在概率多项式时间内实现。

命题 4.4.10：假定在构造 4.4.9 的最后一步中使用的子协议实际上是一个 $G3C$ 的辅助输入零知识证明，那么构造 4.4.9 构成 L 的一个辅助输入零知识证明。

证明：构造 4.4.9 构成 L 的一个交互证明这个事实可以立即从约简的有效性（和它使用了 $G3C$ 的交互证明这个事实）得到。乍看起来，构造 4.4.9 的零知识特性似乎很容易得到。然而，存在一个不容忽视的小问题：在构造 4.4.9 中调用的 $G3C$ 零知识证明中的验证者，不仅具有公共输入图 G，还具有可以约简为 G 的原始公共输入 x。如果对于辅助输入来说这个证明系统不是零知识的，那么这个附加信息可能会帮助验证者在 $G3C$ 交互证明中获取知识。细节如下。

假设我们需要模拟机器 V^* 与构造 4.4.9 的证明者在公共输入 x 下的交互。不失一般

性，我们可以假设机器 V' 调用交互机 V''，而交互机 V'' 与 G3C 交互证明中具有公共输入 $G=f_1(x)$ 和辅助输入 x 的证明者交互。根据 G3C 交互证明是辅助输入零知识的这一假设，可知存在对于输入 (G,x)，模拟 V'' 与 G3C 证明者的交互（对于公共输入 G 和验证者的辅助输入 x）的模拟器 M''。因此，构造 4.1.9 的模拟器 M' 如下操作：对于输入 x，模拟器 M' 计算 $G\stackrel{\text{def}}{=}f_L(x)$，并输出 $M''(G,x)$。命题成立。■

解决前面提到的小问题的另一个方法是，函数 f_l（即，通过标准约简得到的函数）可以在多项式时间内进行求逆（参考习题 17）。在任何情况下，我们都可以立即得到如下定理。

定理 4.4.11：假定存在一个承诺方案满足（非均匀）保密性要求和明确性要求。那么 ***NP*** 中的每种语言都具有辅助输入零知识证明系统。此外，假如能够得到相应的 ***NP*** 证据作为辅助输入，那么这个系统中指定的证明者可以在概率多项式时间内实现。

我们要提醒读者，上面定理的条件满足当（且仅当）存在（非均匀）单向函数：参考定理 3.5.12（单向函数隐含伪随机发生器）、命题 4.4.5（伪随机发生器隐含承诺方案）以及习题 13（承诺方案隐含单向函数）。

应用：一个实例

定理 4.4.11 的一个典型应用是，可以证明秘密的一些特性而不泄露该秘密。具体地说，考虑一方（记为 S）向另一方（记为 R）进行承诺。假设在后面的阶段中，S 将会公布承诺数值的部分信息，但不会公布全部信息。例如，S 可能会公布一个单独的比特，这个比特用来指定承诺数值是否大于由 R 指定的某个数值。如果 S 只是将这个比特发送给 R，那么 R 将不知道接收到的位是否是正确的。使用零知识证明可以使 S 能够让 R 确信已经公布的比特是正确的，而又不产生任何附加的知识。这样的零知识证明的存在性可以根据定理 4.4.11 得到，并且要被证明的命题是 ***NP*** 类型的（S 知道相应的 ***NP*** 证据）。

并不完全确信前述论题（即，定理 4.4.11 的应用）有效性的读者，可以形式化描述如下：令 v 为 S 对其进行承诺的数值，s 为在承诺阶段中使用的随机性，$c\stackrel{\text{def}}{=}C_s(v)$ 为得到的承诺（相应于承诺方案 C）。假设 S 想要向 R 证明，c 是大于 u 的数值的承诺。因此，S 想要（用零知识）证明的是，存在 v 和 s，使得 $c=C_s(v)$ 并且 $v>u$，其中 c 和 u 对于 R 是已知的。实际上，这是一个 ***NP*** 类型的命题，S 知道相应的 ***NP*** 证据（即 (v,s)），因为是它自己选择的 v 和 s。最后，我们定义一个语言

$$L \stackrel{\text{def}}{=} \{(c,u) : \exists v,s\ s.t.\ c = C_s(v) \text{ 且 } v > u\}$$

显然，语言 L 是属于 ***NP*** 的，并且对于 $(c,u)\in L$ 的 ***NP*** 证据是 (v,s) 对。因此，定理 4.4.11 适用。

在习题 18 中举了另一些实例。其他的应用将会在第 2 册中列举。

我们强调，因为它是一个一般（在一定意义上是普遍的）的结论，不能指望定理 4.4.11 中的构造能够提供应用解决方案（特别是在应用比较简单时）。定理 4.4.11 应该被认为是一个善辩命题：它证明的是，存在很大一类原则上可以解决的密码问题（这些问题相当于根据一些公共信息来证明保密活动的相容性）。因此，当在实际中面对这样的问题时，可以推断存在一个解决方法。这只是第一步，接下来是寻找一个可行的方法。

IP **中语言的零知识**

有趣的是，定理 4.4.11 的结果可以在“相同的条件下每种具有交互证明系统的语言同时也具有零知识交互证明系统”的意义上，扩展到“最大”。也就是说：

定理 4.4.12：假设存在一个承诺方案，满足（非均匀）保密性要求和明确性要求，那么 **IP** 中的每种语言都具有零知识证明系统。

我们相信这个扩展（从定理 4.4.11 到定理 4.4.12 的扩展）并不会有很大的实际意义。通过先将 L 的交互证明转换为具有良好完备性的公共硬币交互证明（见 4.2.3 小节），可以证明定理 4.4.12。在后面的证明系统中，假设验证者发送随机字符串（不考虑前面的证明者的消息），并通过对通信过程使用某个多项式时间预测来决定是否要接受。因此，我们可以通过让新的证明者发送原来（公共硬币系统）证明者发送消息的承诺，而不是发送这些消息的明文，来修改这个证明系统。一旦这个“加密的”交互完成，证明者就以零知识的方式证明了原来的验证者一定会接受这个隐藏的副本（这是 **NP** 中的一个命题）。因此，定理 4.4.12 就可以用定理 4.4.11 来证明。

4.4.4 二级考虑

当为 **NP** 中的每种语言给出零知识证明系统时，我们并没有试图给出可能的最高效的构造。我们最关心的是要给出尽可能简单而又能说明问题的证明。然而，一旦我们知道 **NP** 零知识证明是存在的，那么自然就会问这些零知识证明的效率怎样。更为重要的是，我们引入并讨论一个更为精练的零知识证明的“实际保密性”的度量，称为知识紧密性。

为了建立比较零知识证明的通用基础，我们需要指定这些证明的错误概率的一个合适的度量。将在下面使用的一个指导性选择是，考虑具有错误概率 2^{-k} 的零知识证明的复杂度，其中 k 是依赖于公共输入长度的一个参数。当比较零知识证明时，另一个需要记住的问题就是这些证明成立的假设条件。我们在本节中坚持使用迄今为止一直使用的假设（即，单向函数的存在性）。

4.4.4.1 标准的效率度量

将要考虑的自然、标准的效率度量如下：

- 证明的通信复杂度。最重要的通信度量就是交互复杂度（即消息交换的数目）。在交互中交换的总比特数也是一个重要的考虑因素。
- 证明的计算复杂度（具体地，每一方所采用的基本步骤数）。

只要通信复杂度和计算复杂度之间的折衷是合理的，通信复杂度看起来就比计算复杂度更为重要。

为了说明这些度量，我们考虑构造 4.4.7 中给出的 $G3C$ 零知识证明。回想一下，这个证明系统对公共输入图 $G=(V,E)$，具有一个中等程度的接受间距，具体来说是 $1/|E|$。因此，为了得到具有错误概率 e^{-k} 的零知识证明，需要连续应用 $k\cdot|E|$ 次构造 4.4.7，其中 $e\approx 2.718$ 是自然对数基。因此，得到的零知识证明的交互复杂度为 $O(k\cdot|E|)$，比特复杂度为 $O(k\cdot|E|\cdot|V|^2)$，计算复杂度为 $O(k\cdot|E|\cdot\text{poly}(|V|))$，其中多项式 $\text{poly}(\cdot)$ 依赖于实际使用的承诺方案。

可以为 **NP** 中特定的语言制定更为有效的零知识证明系统。而且，即使采用将 **NP** 中语言的零知识证明系统的构造约简为单个 **NP** 完备语言的零知识证明系统的方法，也可以得到很大的改进。例如，使用习题 20，就可以得到具有交互复杂度 $O(k)$、比特复杂度 $O(k\cdot$

$|V|^{2-\epsilon}$)和计算复杂度 $O(k \cdot |V|^{2+O(\epsilon)})$的哈密尔顿环路问题(还是有 e^{-k}的错误概率)的零知识证明,其中 $\epsilon>0$ 是一个与承诺方案要求的保密性相关的常量(在构造 4.4.7 和习题 20 中我们取 $\epsilon=1$)。要注意的是,与实例大小相关的复杂度受到问题之间约简的影响,因此,公平的比较是考虑一般问题(即,有限终止(Bounded Halting))的复杂度。

协议的交互复杂度对效率有重要的影响,因此希望尽量减小该复杂度。特别地,希望零知识证明具有固定数目的交互和可忽略的错误概率。在 4.9 节中将会继续追寻这个目标。

4.4.4.2 知识紧密性

前述的效率度量是通用的,因为它们适用于任何协议(与是不是零知识无关)。因为安全性和效率经常是可以互相转化的(特别是在现在的环境中),应该与安全性的精练度量结合考虑效率的精练度量。

与通用的(效率)度量不同,我们只考虑零知识的(安全性)度量,称为知识紧密性。直观地说,知识紧密性是零知识的一个精练,其目标在于衡量证明系统的"真正安全性",即,当不与证明者交互时,验证者要工作到怎样的程度才能完成它在与证明者交互后才能完成的计算。因此,知识紧密性是模拟器(期望)的运行时间与模拟器模拟的真实交互中验证者所用的运行时间之间的比率。值得注意的是,迄今为止所给出的模拟器以及所有已知的模拟器都是通过重复随机实验来操作,因此,紧密性的一个有效的度量应该考虑它们的期望运行时间(假设它们从不产生错误,即不会输出特殊符号⊥),而不是最坏情况下的运行时间(换句话说,可以考虑以不超过 $\frac{1}{2}$ 的概率输出⊥的模拟器的运行时间)。

定义 4.4.13(知识紧密性):令 $t:\mathbb{N}\to\mathbb{N}$ 是一个函数。我们说语言 L 的零知识证明具有知识紧密性 $t(\cdot)$,如果存在一个多项式 $p(\cdot)$,使得对于每个概率多项式时间验证者 V^*,都存在一个模拟器 M^*(如定义 4.3.2),满足对于所有足够长的 $x\in L$,我们都有

$$\frac{\mathrm{Time}_{M^*}(x)-p(|x|)}{\mathrm{Time}_{V^*}(x)}\le t(|x|)$$

其中,$\mathrm{Time}_{M^*}(x)$代表 M^* 在输入为 x 时的期望运行时间,而 $\mathrm{Time}_{V^*}(x)$表示 V^* 在输入为 x 时的运行时间。

我们假设这样一个计算模型,允许一台机器模仿另一台机器,但是只消耗被模仿机器的运行时间。前面定义中的多项式 $p(\cdot)$的目的是处理由模拟产生的一般开销(只有在验证者 V^* 非常快的情况下才是重要的)。我们指出,零知识的定义并不能保证知识紧密性是多项式的。然而,所有已知的零知识证明,更一般地说,所有用具有访问 V^* 的黑盒子的单个模拟器的零知识特性,都具有多项式知识紧密性。特别地,构造 4.3.8(与习题 20 中的构造相似)具有知识紧密性 2,而构造 4.4.7 具有知识紧密性大约 $\frac{3}{2}$。我们相信知识紧密性是一个很重要的效率考虑因素,而且希望它是一个常量。

我们指出,知识紧密性的概念在与下面的论断协调时也是有指导性的:

1. 并行执行 $O(\log n)$次构造 4.4.7,其中 n 是图中的顶点数,会产生零知识证明系统。

并行执行多于 $O(\log n)$次(比如说,$O(\log n)\cdot(\log\log n)$)次)构造 4.4.7,不知道是否会产生零知识证明系统。(而且,证明得到的结果系统是零知识的是不可能的,参考 4.5.4.2 节。)

一旦意识到并行执行 $k(n)=O(\log n)$次构造 4.4.7,将会产生一个知识紧密性大约为 $(3/2)^{k(n)}$ 的零知识证明系统,这些矛盾的命题之间的间距会显得不那么明显(见习题 19)。

4.5* 否定结果

在这一节里，我们回顾一下零知识的一些否定结果。这些结果显示，在前述章节中所提到的结果和构造的一些缺陷是不可避免的。最重要的是，定理 4.4.11 断言，如果单向函数存在，那么 **NP**(计算)零知识交互证明系统也存在。自然就产生了如下 3 个问题：

1. 无条件结果：在不做任何假设的情况下，能否证明 **NP**(计算)零知识证明系统的存在性？

2. 完备零知识：即使在某些合理假设的条件下，能否给出 **NP** 完备零知识证明系统？

3. 随机性和交互的作用：例如，能否提出无差错的 **NP** 零知识证明系统？

这些问题的答案看起来好像都是否定的。

关于零知识证明的另一个重要问题是零知识证明在并行合成运算下的保持性。我们将指出，一般来说，在并行合成运算下并不保持零知识特性(即，存在一对零知识协议，当并行执行这个协议对的时候将会泄露知识)。而且，我们将考虑一些由零知识证明的并行合成得到的自然证明系统，并且说明所产生的合成证明不能被证明是零知识的。

结构安排　我们首先回顾一些与定理 4.4.11 中随机性和交互的基本作用相关的结果(即，**NP** 零知识证明的存在性)。对于这些结果我们也提出一些相对简单的证明思想(参见 4.5.1 小节)。然后，在 4.5.2 小节中，我们将介绍 **NP** 零知识证明的存在性意味着某种形式的平均单向困难性，因此定理 4.4.11 中的假设不能完全取消。在 4.5.3 小节中，我们介绍完备零知识证明系统，在 4.5.4 小节中，我们介绍零知识协议的合成。我们指出，本节中除了定理 4.5.8(即，完备零知识证明的限制)外的所有结果都适用于 4.8 节中将要定义和讨论的零知识理论。

4.5.1　交互和随机性的重要性

如果一个证明系统是 **BPP** 中某种语言的，那么我们说这种证明系统是平凡的。因为在验证者同证明者没有任何交互时，**BPP** 中的语言可由验证者决定，所以这些证明系统是毫无用处的(至少对于密码学来说是如此)。

单向零知识证明的平凡性　单向证明系统是只有一个消息被发送(即，从证明者到验证者)的证明系统。我们指出，这种包含 **NP** 类型证明作为特殊情形的特殊交互证明系统类太严格了，以致于不能包含非平凡零知识证明。

定理 4.5.1：如果 L 具有单向零知识证明系统，那么 $L\in$ **BPP**。

证明思路：给定一个系统中诚实验证者观点的模拟器 M(如定义 4.3.3 中所规定的)，我们构造一个 L 的判决过程。在输入 x 下，我们调用 $M(x)$ 并且得到 (ω,r)，这里假定 ω 是证明者发送的消息，$r\in\{0,1\}^t$ 假定为验证者的随机磁带。我们均匀选择 $r'\in\{0,1\}^t$，并且就像真正的验证者那样通过接收消息 w 和将 r' 作为随机磁带的内容来判决。对于 $x\in L$ 这种情况的分析，使用 M 是一个好的模拟器的假设；而对于 $x\notin L$ 的情况的分析，使用证明系统的合理性(和 r' 是独立于 w 来选择的这个事实)。

验证者随机性的基本作用　下面我们说明验证者的随机性对于零知识证明系统的非平凡性来说是必要的。可以得到，非零错误概率对于零知识证明系统的非平凡性来说是本质的，因为如果不是如此的话，验证者总是可以将它的随机磁带全设为零。（实际上，我们能够直接证明非零合理性错误对于零知识证明系统的非平凡性是必不可少的，并且可以证明定理 4.5.2 是一个特例[①]。）

定理 4.5.2：假设 L 有一个零知识证明系统，在这个系统中验证者的程序是确定性的，那么 $L\in$ ***BPP***。

证明思路：因为验证者是确定性的，所以证明者能够完全决定验证者以后发送的每个消息。这样证明系统就转化成一个与其等价的证明系统，在这个等价的证明系统中，证明者一次性发送给验证者整个执行过程的副本。注意，原证明系统的完备性、合理性和零知识特性仍然保留，并且所得出的结果证明系统是单向的。应用定理 4.5.1，我们就得到了结论。

证明者随机性的基本作用　最后，我们来说明证明者一方的随机性对于零知识系统的非平凡性也是必要的。

定理 4.5.3：假定 L 有辅助输入零知识证明系统，在这个系统中，证明者程序是确定的，那么 $L\in$ ***BPP***。

注意到这里的假设（即零知识类型的要求）更强。（计算无限制的确定性证明者对于零知识基本定义（即定义 4.3.2）的非平凡性是足够的。）

证明思路：不失一般性，假定验证者是证明系统中发送第一个消息的一方。我们考虑一个具有欺骗企图的验证者，这个验证者在给定辅助输入 $z_1,\ldots,z_t$ 时，发送 z_i 作为第 i 个消息。该验证者的其余消息任意决定。我们首先注意到，因为证明者是确定性的，那么在真实交互中，前 $i(i\leq t)$ 个证明者的应答由 $z_1,\ldots,z_i$ 决定。因此在模拟中也应该这样。我们通过模拟在公共输入 x 下，指定的证明者和指定的验证者之间的交互来构造一个判决过程，该过程的输入也为 x。进而，我们均匀选择和固定一个验证者的随机磁带，记为 r。该模拟过程反复进行并对证明者的消息应答。为了获得证明者的下一条消息，我们首先确定验证者的下一条消息（通过对输入 x、硬币 r 和当前为止接收到的消息运行指定验证者的程序）。然后，对于输入 $(x,(z_1,\ldots,z_i))$，我们调用模拟器，其中 $z_1,\ldots,z_i$ 是到目前为止确定的验证者消息，因此我们得到并且记录下了证明者的第 i 个消息。我们的最终判决由验证者的判决决定。

4.5.2　无条件结果的限制

定理 4.4.12 说明：如果非均匀单向函数存在，那么 ***IP*** 中的所有语言都存在零知识证明。在本小节中我们来考虑这个充分条件是否也是必要条件的问题。下面的一些结果似乎提供了一些这一方向上的（然而，是不明显的）线索。特别地，***BPP*** 之外语言的零知识证明系统的存在性暗示了（弱）单向性。用对偶的方式，那些很难去近似的语言的零知识证明系统的存在性（在

① 给定一个系统中诚实验证者观点的模拟器 M，我们构造一个关于 L 的判决过程。对于输入 x，我们调用 $M(x)$，并且当且仅当它的输出与诚实验证者将会接受的输出副本一致时接受。对于 $x\in L$ 这种情况的分析，使用 M 是一个好的模拟器的假设；而对于 $x\notin L$ 情况的分析，使用证明系统的完备合理性。定理 4.5.2 成立是因为确定的验证者必须具有零合理性差错。

平均意义上)暗示了单向函数(而不是非均匀单向函数)的存在性。在本小节的其他部分,我们来精确叙述一下这些结果。

零知识的非平凡性隐含弱单向性　我们所说的零知识的非平凡性是指 ***BPP*** 之外语言的零知识证明系统的存在性(因为后者(***BPP***)具有平凡的零知识证明系统,在该系统中证明者什么都不做)。下面我们来澄清一下什么是"弱单向性"。我们的出发点是单向函数类(即定义 2.4.3)的定义。回想一下,这些函数类都以某个 $\bar{I}\subseteq\{0,1\}^*$ 为下标,并且容易取样和评估,但是通常难于求逆。也就是说,一个典型函数 $f_i(i\in\bar{I})$ 在已知一个典型的像时求逆很难。这里我们只要求这个类中存在对于一个典型的像难以求逆的函数。

定义 4.5.4(具有单向实例的函数类):函数类 $\{f_i:D_i\to\{0,1\}^*\}_{i\in\bar{I}}$ 被称为**具有单向实例**,如果存在 3 个概率多项式时间算法 I,D 和 F,满足如下两个条件:

1. 容易采样和计算:如定义 2.4.3 所述。

2. 某些函数难以求逆:对于每个概率多项式时间算法 A'、每个多项式 $p(\cdot)$,和无穷多个 $i\in\bar{I}$,都有

$$\Pr[A'(i,f_i(X_i))\in f_i^{-1}(f_i(X_i))]<\frac{1}{p(|i|)}$$

其中 $X_i=D(i)$。

实际上,因为困难性条件并不涉及由 I 导出的分布,所以我们可以将 I 从定义中忽略,而只考虑下标集 $\bar{I}$。这样的函数类包含无穷多个难以求逆的函数,但是可能无法找到有效的方式来选择这样的函数(因此,这样的函数类就没有真正意义了)。还有,我们强调困难性条件具有平均意义;这些无穷多的函数中的每一个在很强的概率意义上都是难以求逆的,而不是指每一个都完全是不可逆的。

定理 4.5.5:如果存在 ***BPP*** 之外语言的零知识证明,那么存在具有单向函数实例的函数类。

我们指出,仅有 ***BPP***$\subset$***IP*** 的假设,并不知道是否隐含(平均)单向性。甚至 ***NP*** 中而不是 ***BPP*** 中语言的零知识证明也没有隐含平均意义上的困难性;它只是隐含了在最坏情况下,易于计算而难于求逆的函数的存在性(参见 2.1 节)。

产生单向函数的"困难"语言的零知识　我们的困难语言的概念如下。

定义 4.5.6:我们说语言 L 是**难于近似的**,如果存在一个概率多项式时间算法 S,使得对于每个概率多项式时间算法 A、每个多项式 $p(\cdot)$ 以及所有足够大的 n,都有

$$\Pr[A(X_n)=\chi_L(\chi_n)]<\frac{1}{2}+\frac{1}{p(n)}$$

其中 $X_n\stackrel{\text{def}}{=}S(1^n)$,$\chi_L$ 是语言 L 的特征函数(即,如果 $x\in L$,则 $\chi_L(x)=1$,否则 $\chi_L(x)=0$)。

例如,如果 f 是一个单向置换,b 是 f 的一个核心断言,那么语言

$$L_f\stackrel{\text{def}}{=}\{x\in\{0,1\}^*:b(f^{-1}(x))=1\}\in \boldsymbol{NP}$$

(在均匀分布下)是难以近似的。

定理 4.5.7:如果对于难以近似的语言存在零知识证明,那么就存在单向函数。

我们强调,只有难以近似语言的存在性并不能保证隐含单向函数的存在性(参见 2.1 节)。

4.5.3 统计零知识证明的限制

下面我们会给出具有完备零知识证明系统的语言类的界限定理。事实上，这个界限甚至涉及到了统计(即几乎完备)零知识证明系统(参见 4.3.1.4 小节)。我们先给出一些背景知识。我们用 AM 表示具有按如下步骤操作的交互证明的语言类。首先验证者发送给证明者一个随机字符串，然后证明者用某个字符串应答，最后验证者用一个(依赖于公共输入及这两个字符串)确定性计算来决定是接受还是拒绝。我们相信 $\mathbf{AM}$ 不包含 co$\mathbf{NP}$(或者 co$\mathbf{AM}$ 不包含 $\mathbf{NP}$)。co$\mathbf{NP}\subseteq\mathbf{AM}$ 暗示着多项式时间体系结构的消耗这个事实也是对上面结论的额外支持。不管怎样，我们所要介绍的结论都按如下形式表述.

定理 4.5.8：如果语言 L 存在一个统计(几乎完备)零知识证明系统，那么 $L\in$ co$\mathbf{AM}$(事实上，$L\in$ co$\mathbf{AM}\cap\mathbf{AM}$)。

在统计零知识的一些限制有所放松的情况下(例如，允许模拟器以期望多项式时间运行)，这条定理仍然成立。因此，如果某种 $\mathbf{NP}$－完全语言有一个统计零知识证明系统，那么 co$\mathbf{NP}\subseteq\mathbf{AM}$，这是不可能的。

我们要强调，定理 4.5.8 并不能应用于 4.8 节中定义和讨论的完备(或统计)零知识理论。因此，在定理 4.5.8 与某些合理复杂性假设的条件下，$\mathbf{NP}$ 中的每种语言都确实存在完备零知识理论这个事实不矛盾。

4.5.4 零知识和并行合成

我们给出两个关于零知识协议并行合成的否定结果。这两个结果就其概念的立足点来说是大不相同的：第一个结果断言并行合成猜想(一般情况下)不成立(即并行运行两个零知识协议将会产生一个零知识协议的猜想)，却只字未提特定的自然的例子。第二个结果谈到了一个交互证明类，这个交互证明类包括一些有趣的和自然的例子，然而它断言使用一般的证明规范(通常称作"黑盒模拟器")并不能证明这个类的成员是零知识的。第二个结果与本小节的关系可以从后面这个事实得到，即这个类中的某些成员可以通过自然零知识证明的并行合成得到。我们指出，(不通过提供黑盒模拟器的方式)寻找一种描述协议的零知识特性的方式是很难的。

我们强调，"并行合成"是指并行运行协议的几个副本，在执行协议的过程中指定的(诚实)参与方独立地运行自己的副本。特别地，如果需要某一方在某一轮中掷硬币，那么它应该对每个副本分别独立地投掷硬币。

4.5.4.1 并行合成猜想的失败

作为信任不合理的直觉的警告，我们指出，多年以来(参考零知识证明引言)很多研究者都坚持下面的猜想是正确的。

并行合成猜想：令 P_1 和 P_2 为两个零知识证明者，那么通过并行运行 P_1 和 P_2 所产生的证明者也是零知识的。

然而，并行合成猜想是不成立的。

命题 4.5.9：存在两个均为零知识的证明者 P_1 和 P_2，使得由并行运行 P_1 和 P_2 所生成的

证明者会产生知识(例如,一个欺骗性的验证者可以从这个证明者身上获得某个多项式时间内不能解决的问题的解法)。而且,即使任意一个 P_i 的零知识特性可通过使用验证者作为黑盒子(同定义 4.5.10)的模拟器来描述,上述结论仍然成立。

证明思路:考虑一个证明者,记为 P_1,当且仅当验证者能回答某个随机选择的难题(即,我们强调这个问题是由 P_1 来选择的)时才发送"知识"给验证者。这些难题的答案看起来是伪随机的,然而 P_1(P_1 在计算上不受限制)能够验证它们的正确性。现在考虑另一个(计算不受限制的)证明者,记为 P_2,P_2 回答这些难题。这两个证明者(本身)都是零知识的:P_1 是零知识的是因为任何概率多项式时间验证者都不能回答它的问题,而 P_2 是零知识的是因为它的应答可通过随机字符串来模拟。但是,一旦它们并行运行,一个欺骗性的验证者可以把 P_1 的问题发给 P_2,然后从 P_2 的回答中获得 P_1 的知识。为了将这个思路转化成证明,我们需要用前述假设特性来构造一个困难问题。

上述命题说明了具有指数时间证明者的并行合成猜想不成立。假定单向函数存在,则对于(带有辅助输入的)概率多项式时间证明者的并行合成猜想也是不成立的。例如,考虑利用知识证明(参见 4.7 节)的两个证明者 P_1 和 P_2。令 C 为一个比特承诺方案(因为单向函数存在,所以我们知道这个比特承诺方案是存在的)。对于公共输入 $C(1^n,\sigma)$,其中 $\sigma\in\{0,1\}$,证明者 P_1 以零知识的方式向验证者证明它知道 σ(这里,将承诺中使用的硬币作为证明者的辅助输入)。与此相反,对于公共输入 $C(1^n,\sigma)$,证明者 P_2 要求验证者证明它知道 σ,并且如果 P_2 相信了验证者,那么 P_2 将 σ 发送给验证者。这个验证者与 P_1 使用相同的知识证明系统。显然,每个证明者都是零知识的,但它们的并行合成却不是零知识的。

同样,使用更强的困难性假设,我们能证明几乎完备零知识(而不是计算零知识)的并行合成猜想也不成立(这里我们令证明者使用完备零知识、计算合理的知识证明;参见 4.8 节)。

4.5.4.2 "自然"零知识带来的问题

根据定义,为了说明证明者是零知识的,我们必须对每个期望验证者 V^* 都提出相应的模拟器 M^*(M^* 模拟 V^* 和证明者的交互)。然而,所有已知的零知识描述都通过提出一个"通用"模拟器来实现,这个"通用"模拟器将任何期望验证者 V^* 作为黑盒来使用。实际上,这些描述将验证者程序(即 V^*)确定的下一个消息函数、这个函数的辅助输入和随机输入作为黑盒(模拟器的这个属性隐含在前面章节中模拟器的构造中)。我们注意到,很难构造出一种描述零知识特性的可选方式(因为将验证者作为非黑盒来使用似乎要求代码能够"反设计(reverse engineering)")。在辅助输入零知识的情况下,这种难度更大。

定义 4.5.10(黑盒零知识):

- 下一个消息函数:令 B 为交互图灵机,令 x,z,r 分别为表示公共输入、辅助输入和随机输入的字符串。考虑描述机器 B 所发消息的函数 $B_{x,z,r}(\cdot)$,$B_{x,z,r}(\overline{m})$表示在公共输入为 x、辅助输入为 z、随机输入为 r 以及接收的消息序列为 $\overline{m}$ 时,B 所发送的消息。为了简单起见,我们假定 B 的输出就是它的最后一条消息。
- 黑盒模拟器:我们说概率多项式时间预言机 M 是对于证明者 P 和语言 L 的黑盒模拟器,如果对于每个多项式时间交互机 B、每个概率多项式时间预言机 D、每个多项式 $p(\cdot)$、所有足够大的 $x\in L$ 和每个 $z,r\in\{0,1\}^*$,都有

$$\left|\Pr[D^{B_{r,x,z}}(\langle P,B_r(z)\rangle(x)) = 1] - \Pr[D^{B_{r,x,z}}(M^{B_{r,x,z}}(x)) = 1]\right| < \frac{1}{p(|x|)}$$

其中 $B_r(z)$ 表示与具有辅助输入 z 以及随机输入 r 的 B 的交互。

若 P 有黑盒模拟器，我们就说 P 是**黑盒零知识的**。

从本质上说，这个定义是说，黑盒模拟器模拟了证明者 P 与多项式时间验证者 B 的交互，其中 B 是可以得到辅助输入(即 z)和选择随机输入(即 r)的。验证者仅仅通过对 $B_{x,z,r}$($B_{x,z,r}$ 指定了在输入为 x、辅助输入为 z、随机输入为 r 时，B 发送的下一条消息)的调用就可以(高效地)实现这些功能。即使在分辨算法 D 可以访问预言 $B_{x,z,r}$ 时，模拟过程与真实的交互过程也是不可分辨的。习题 21 中给出了一个等价公式。显然，如果 P 是黑盒零知识的，那么它也是辅助输入零知识的(并且具有以多项式为界的知识紧密性，参见定义 4.4.13)。

定理 4.5.11：假设(P,V)是一个语言 L 的具有可忽略错误概率的交互证明系统。再假设(P,V)具有如下特性：

- 常数轮数(constant round)：存在一个整数 k，使得对于每个 $x \in L$，当输入为 x 时，证明者 P 至多发送 k 条消息。
- 公开硬币：验证者 V 发送的消息由它的随机磁带的连续片段预先确定。
- 黑盒零知识：证明者 P 有一个(在语言 L 上的)黑盒模拟器。

那么 $L \in$ ***BPP***。

这个定理对于 4.8 节中定义和讨论的计算合理零知识证明系统仍然成立。

我们注意到，构造 4.3.8(图同构零知识证明)和构造 4.4.7(图着色零知识证明)都是常数轮数的，都是使用公开硬币的，而且(对于相应的语言)都是黑盒零知识的。然而它们确实没有可忽略错误概率。但是，并行运行每个构造多项式次就可生成相应语言的具有可忽略错误概率的交互证明[①]。显然所生成的证明系统也是常数轮数且使用公开硬币的。因此，除非相应的语言在 ***BPP*** 中，否则所生成的证明系统就不是黑盒零知识的。 14 反

定理 4.5.11 有时候被解释为就是用来说明具有公开硬币(又名 *Arthur-Merlin* 游戏)的交互证明的先天缺陷的。这样的证明既不可能是轮数有效的(即，具有常数轮数和可以忽略的错误)，也不可能是黑盒零知识的(除非相应语言在 ***BPP*** 中)。换句话说，当需要构造轮数有效(对于不在 ***BPP*** 中的语言)的零知识证明系统的时候，我们应该使用“私有硬币”(即，验证者根据硬币投掷来发送消息，但并不公布硬币投掷的结果)。这实际上就是 4.9 节中要采用的方法。

4.6* 证据不可分辨性和隐藏性

由于在并行合成运算下零知识是不封闭的(见 4.5.4 小节)，所以在并行合成运算下封闭的“秘密”准则就具有重要的实践意义和理论意义。本节将介绍关于(***NP*** 中语言的)交互证明

---

① 实际上，超对数次的重复对于构造 4.3.8 中的情况和构造 4.4.7 的修改版本就足够了。修改的证明系统通过对由输入图通过应用由所谓的 PCP 定理确保的多项式时间约简方法得到的图调用构造 4.4.7 而得到。特别地，这种约简方法将 G3C 约简为 G3C，而 G3C 之外的图被映射为这样的图：项点集的每个三向分割都至少有一个常量违反边(即，两个顶点都在同一个分割中的边上)比例。令 $\epsilon > 0$ 是由 PCP 定理指定的常量。那么，得到的证明系统具有至多为 $1-\epsilon$ 的极好完备性和合理性错误，因此，超对数次的重复将产生可忽略错误概率。

系统的"秘密性"的两个概念,证据不可分辨性和证据隐藏性。这两个概念看起来要比零知识弱一些,但是对于某些特定的应用来说已经足够了。

我们指出,像零知识一样,证据不可分辨性和证据隐藏性都是证明者的特性(更一般地说,都是交互机的特性)。

4.6.1 定义

在本节中,我们只讨论 **NP** 中语言的证明系统。注意,语言 $L\in \mathcal{NP}$ 的证据关系是一个多项式限定的(即,若$(x,y)\in R_L$,那么$|y|\leq \text{poly}(|x|)$)二元关系 R_L,是多项式时间可分辨的,通过下式刻划 L

$$L=\{x:\exists\, y \text{ s.t. } (x,y)\in R_L\}$$

对于 $x\in L$,任何满足$(x,y)\in R_L$ 的 y 都称为(对成员关系 $x\in L$ 的)证据。我们令 $R_L(x)$ 表示成员关系 $x\in L$ 证据的集合,即 $R_L(x)\stackrel{\text{def}}{=}\{y:(x,y)\in R_L\}$。

4.6.1.1 证据不可分辨性

不严格地说,语言 $L\in \mathcal{NP}$ 的交互证明是证据独立的(也称证据不可分辨的),如果验证者对与证明者交互过程的观点和证明者的辅助输入是统计独立的(也称"计算独立")。实际上,我们会将这些要求特殊化为如下情形:辅助输入构成了公共输入的 **NP** 证据;即,对于语言 $L\in \mathcal{NP}$ 的证据关系 R_L,我们仅仅考虑在公共输入 $x\in L$ 时的交互,其中证明者被给定 $R_L(x)$中的辅助输入。观点与证据是计算独立的是指:对于每两个辅助输入,得到的观点都是计算独立的。与 4.3 节中的讨论相似,通过考虑与证明者交互时验证者的观点或者验证者在交互之后的输出,我们可以得到等价的定义。在实际的定义中,我们采用后一种(即交互后验证者的"输出")公式表达,并使用定义 4.3.10 中的符号。

定义 4.6.1(证据不可分辨性/独立性):令(P,V),$L\in \mathcal{NP}$ 及 V^* 如定义 4.3.10 中定义的一样,令 R_L 为语言 L 的固定证明关系。我们说(P,V)对于 R_L 是**证据不可分辨的**,如果对于每个概率多项式时间交互机 V^* 和每两个序列 $W^1=\{w_x^1\}_{x\in L}$ 和 $W^2=\{w_x^2\}_{x\in L}$,都使得 $w_x^1,w_x^2\in R_L(x)$,并且如下两个总体是计算独立的:

- $\{\langle P(w_x^1),V^*(z)\rangle(x)\}_{x\in L,z\in\{0,1\}^*}$
- $\{\langle P(w_x^2),V^*(z)\rangle(x)\}_{x\in L,z\in\{0,1\}^*}$

也就是说,对于每个概率多项式时间算法 D、每个多项式 $p(\cdot)$、所有足够长的 $x\in L$ 和所有的 $z\in\{0,1\}^*$,都有

$$|\Pr[D(x,z,\langle P(w_x^1),V^*(z)\rangle(x))=1]$$
$$-\Pr[D(x,z,\langle P(w_x^2),V^*(z)\rangle(x))=1]|<\frac{1}{p(|x|)}$$

如果前述的总体是同分布的,我们就说(P,V)对于 R_L 是**证据独立的**。也就是说,对于每个 $x\in L$、每个 $w_x^1,w_x^2\in R_L(x)$和 $z\in\{0,1\}^*$,随机变量$\langle P(w_x^1),V^*(z)\rangle(x)$和$\langle P(w_x^2),V^*(z)\rangle(x)$是同分布的。

特别地,z 可以等于(w_x^1,w_x^2)。下面是一些附加注释:

- 证明者忽略辅助输入的证明系统是(平凡)证据独立的。特别地,指数时间证明者可以

忽略它们的辅助输入(而不会降低它们使验证者信服的概率),因此可以是平凡证据独立的。然而,概率多项式时间证明者不能忽略它们的辅助输入(因为如果不这样,证明者就会变得毫无用处)。因此,对于概率多项式时间证明者(对于 **BPP** 之外的语言),证据不可分辨性的要求是非平凡的。

- **NP** 中语言的任何零知识证明系统都是证据不可分辨的(因为对应于每个证据的分布可以用相同的模拟器来近似,细节见后)。同样,完备零知识证明是证据独立的。
- 另一方面,证据不可分辨性并不意味着零知识。特别地,任何具有惟一证据的语言对应的证明系统一般都是证据不可分辨的,但是有可能不是零知识的。例如,对于单向置换 f,考虑刻划所有字符串集合的("非自然")证据关系 $\{(f(w),w):w\in\{0,1\}^*\}$,以及对于公共输入 $f(w)$ 和辅助输入 w,发送 w 给验证者的证明者。
- 可以相对较容易地发现,证据不可分辨性和证据独立性在顺序合成运算下是保持的。在下一小节中,我们将会说明它们在并行合成运算下也是保持的。

扩充概念 证据不可分辨性的扩充概念要求,只要证明系统的公共输入是计算不可分辨的,那么相应的验证者观点也是计算不可分辨的。也就是说,我们如下扩充定义 4.6.1。

定义 4.6.2(强证据不可分辨性):令 (P,V) 和所有其他的符号都如定义 4.6.1 中定义的那样。我们说 (P,V) 对于 R_L 是**强证据不可分辨的**,如果对于每个概率多项式时间交互机 V^* 及每两个可能的集合 $\{(X_n^1,Y_n^1,Z_n^1)\}_{n\in\mathbb{N}}$ 和 $\{(X_n^2,Y_n^2,Z_n^2)\}_{n\in\mathbb{N}}$,使得每个 (X_n^i,Y_n^i,Z_n^i) 都取值于 $(R_L\times\{0,1\}^*)\cap(\{0,1\}^n\times\{0,1\}^*\times\{0,1\}^*)$,那么下面结论成立:

如果 $\{(X_n^1,Z_n^1)\}_{n\in\mathbb{N}}$ 和 $\{(X_n^2,Z_n^2)\}_{n\in\mathbb{N}}$ 是计算不可分辨的,那么 $\{\langle P(Y_n^1),V^*(Z_n^1)\rangle(X_n^1)\}_{n\in\mathbb{N}}$ 和 $\{\langle P(Y_n^2),V^*(Z_n^2)\rangle(X_n^2)\}_{n\in\mathbb{N}}$ 也是计算不可分辨的。

我们强调,$\{(X_n^1,Y_n^1,Z_n^1)\}_{n\in\mathbb{N}}$ 和 $\{(X_n^2,Y_n^2,Z_n^2)\}_{n\in\mathbb{N}}$ 并不要求是计算不可分辨的;这样的要求至少就概率多项式时间证明者关心的问题来说,简化了定义。考虑到 X_n^1 和 X_n^2 同分布的特殊情况(并且对 $\{(X_n^i,Y_n^i,Z_n^i)\}_{n\in\mathbb{N}}$ 没有任何计算要求),定义 4.6.1 可由定义 4.6.2 导出。另一方面,假设单向置换存在,证据不可分辨性并不意味着强证据不可分辨性(见习题 25)。而且,可以容易地说明 **NP** 中语言的零知识证明系统是强证据不可分辨的。

命题 4.6.3:令 (P,V) 是语言 $L\in$ **NP** 的一个辅助输入零知识证明系统,那么 (P,V) 是强证据不可分辨的。

证明思想:使用确保 V^* 的模拟器 M^*,我们可以得到 $E^i\stackrel{\text{def}}{=}\{\langle P(Y_n^i),V^*(Z_n^i)\rangle(X_n^i)\}_{n\in\mathbb{N}}$ 和 $S^i\stackrel{\text{def}}{=}\{M^*(X_n^i,Z_n^i)\}_{n\in\mathbb{N}}$ 对于两个 i 都是计算不可分辨的。因此,如果 E^1 和 E^2 不是计算不可分辨的,那么 S^1 和 S^2 也不是计算不可分辨的。将 M^* 合并入分辨器中,就会得到 $\{(X_n^1,Z_n^1)\}_{n\in\mathbb{N}}$ 和 $\{(X_n^2,Z_n^2)\}_{n\in\mathbb{N}}$ 都不是计算不可分辨的。

4.6.1.2 证据隐藏性

现在我们转向证据隐藏性的概念。直观地说,**NP** 中语言的证明系统是证据隐藏的,如果在与证明者交互后,验证者找到公共输入的 **NP** 证据是不可能的。显然,这样的要求只有在从开始时就不可能找到证据的情况下才能够满足。因为每个 **NP** 语言都具有容易搜寻证据的实例,所以我们就必须考虑特别选定的困难情况的证据搜寻问题。这导致了下面的定义。

定义 4.6.4(困难实例的分布):令 $L\in$ **NP**,且 R_L 是 L 的证据关系。令 $X\stackrel{\text{def}}{=}\{X_n\}_{n\in\mathbb{N}}$ 是满

足 $\mathbf{X}_n$ 在 $L\cap\{0,1\}^n$ 上取值的概率总体。我们说 X 对于 R_L 是困难的，如果对于每个概率多项式时间(证据搜寻)算法 F、每个多项式 $p(\cdot)$、所有足够大的 n 和所有 $z\in\{0,1\}^{\text{poly}(n)}$，都有

$$\Pr[F(X_n,z)\in R_L(X_n)]<\frac{1}{p(n)}$$

例如，如果 f 是一个(等长度且非均匀的)单向函数，那么概率总体 $\{f(U_n)\}_{n\in\mathbb{N}}$ 对于证据关系 $\{(f(w),w):w\in\{0,1\}^*\}$ 是困难的，其中 U_n 在 $\{0,1\}^n$ 上服从均匀分布。

定义 4.6.5(证据隐藏性)：令 (P,V)，$L\in\mathcal{NP}$ 和 R_L 同前述定义中定义的那样。令 $X=\{X_n\}_{n\in\mathbb{N}}$ 是一个 R_L 的困难实例总体。我们说 (P,V) 在实例总体 X 下对关系 R_L 是**证据隐藏的**，如果对于每个概率多项式时间机器 V^*、每个多项式 $p(\cdot)$、所有足够大的 n 和所有 $z\in\{0,1\}^*$，都有

$$\Pr[\langle P(Y_n),V^*(z)\rangle(X_n)\in R_L(X_n)]<\frac{1}{p(n)}$$

其中 Y_n 在 $R_L(X_n)$ 上任意分布。

我们指出，两个保密性准则(即证据不可分辨性和证据隐藏性)之间的关系并不是很明显。然而，(对于 $\mathcal{NP}$)零知识证明也是证据隐藏的(对于任何相应的证据关系和任何困难分布)。我们给出定义 4.6.5 的两个扩展。

1. 如果证明系统 (P,V) 在每个 R_L 的困难实例集合下对 R_L 都是证据隐藏的，那么就可以说 (P,V) 对于 R_L 是普遍证据隐藏的(换句话说，可以仅仅要求在每个有效构造的[1] R_L 的困难实例总体下 (P,V) 对 R_L 是证据隐藏的)。

2. 前述定义的其他形式也很有趣，在这些形式中辅助输入 z 被一个依赖于 X_n 的分布 Z_n 取代。这里我们考虑总体 $\{(X_n,Z_n)\}_{n\in\mathbb{N}}$，其中 X_n 在 $L\cap\{0,1\}^n$ 上取值。如果对于每个概率多项式时间算法 F，$F(X_n,Z_n)\in R_L(X_n)$ 的概率可以忽略，那么这样的总体对于 R_L 是困难的。如果对于每个概率多项式时间验证者 V^*，$\langle P(Y_n),V^*(Z_n)\rangle(X_n)\in R_L(X_n)$ 的概率是可忽略的，那么系统 (P,V) 在 $\{(X_n,Z_n)\}_{n\in\mathbb{N}}$ 下对于 R_L 是证据隐藏的。

4.6.2 并行合成

与零知识证明系统相反，证据不可分辨性证明提供一些在并行合成运算下的健壮性。特别地，如果原始的证明者是多项式时间的，那么证据不可分辨证明系统的并行合成结果是证据不可分辨系统。

引理 4.4.6(证据不可分辨性的并行合成引理)：令 $L\in\mathcal{NP}$ 和 R_L 与定义 4.6.1 中相同，假设 P 为概率多项式时间的，(P,V) 对于 R_L 是证据不可分辨的(也称证据独立的)。令 $Q(\cdot)$ 是一个多项式，P_Q 表示对公共输入 $x_1,\ldots,x_{Q(n)}\in\{0,1\}^n$ 和辅助输入 $w_1,\ldots,w_{Q(n)}\in\{0,1\}^*$ 并行调用 P 的程序 $Q(n)$ 次，并使得在第 i 次调用中 P 会以公共输入 x_i 和辅助输入 w_i 为输入。那么 P_Q 对于

$$R_L^Q\stackrel{\text{def}}{=}\{(\overline{x},\overline{w}):\forall i,(x_i,w_i)\in R_L\}$$

是证据不可分辨的(也称证据独立的)，其中 $\overline{x}=(x_1,\ldots,x_m)$，$\overline{w}=(w_1,\ldots,w_m)$，使得 $m=Q$

---

① 见定义 3.2.5。

(n)，并且对每个 i，$|x_i|=n$。

略证 不管是计算形式还是信息理论形式，都可以通过混合理论得到。我们关注计算形式。为了避免烦琐的符号，我们考虑一个对引理失效的一般性的 n（通过矛盾，可知必然存在有限多个这样的 n，并且存在一个可以对所有 n 进行实际处理的精确理论）。也就是说，假设通过使用验证者程序 V_Q^*，可以分辨 P_Q 在公共输入为 $\overline{x}\in L^m$ 的交互中使用的 $\overline{w}^1=(w_1^1,\ldots,w_m^1)$ 和 $\overline{w}^2=(w_1^2,\ldots,w_m^2)$。那么，对某个 i，程序 V_Q^* 也可以分辨混合证据 $\overline{h}^{(i)}=(w_1^1,\ldots,w_i^1,w_{i+1}^2,\ldots,w_m^2)$ 和 $\overline{h}^{i+1}=(w_1^1,\ldots,w_{i+1}^1,w_{i+2}^2,\ldots,w_m^2)$。重写 $\overline{h}^{(i)}=(w_1,\ldots,w_i,w_{i+1}^2,w_{i+2},\ldots,w_m)$ 和 $\overline{h}^{i+1}=(w_1,\ldots,w_i,w_{i+1}^1,w_{i+2},\ldots,w_m)$，其中当 $j\le i$ 时，$w_j\stackrel{\text{def}}{=}w_j^1$；当 $j\ge i+2$ 时，$w_j\stackrel{\text{def}}{=}w_j^2$。通过构造一个能够分辨与初始证明者 P 的多个交互（其中 P 使用的证据）的验证者 V^*，我们可以得出一个矛盾。细节如下：

程序 V^* 集成程序 P 和 V_Q^*，并通过与实际证明者 P 交互和模拟其他 $m-1$ 次与 P（副本程序）的交互来进行。与 P 的实际交互可以被视为在 V_Q^*（与 P_Q）的交互中的第 $i+1$ 个副本，而模拟的交互是与其他的副本相关的。特别地，除了公共输入 x，机器 V^* 将合适的 i 和序列 $\overline{x}$ 以及 $(w_1,\ldots,w_i,w_{i+2},\ldots,w_m)$ 作为它的辅助输入的一部分。对于每个 $j\neq i+1$，机器 V^* 将 x_j 和 w_j 作为 P 的第 j 个副本的公共输入和辅助输入。机器 V^* 对于公共输入 $\overline{x}$，调用 V_Q^*，并且将它作为与 P_Q 进行虚拟交互的接口。V_Q^* 发出的消息 $\overline{\alpha}=(\alpha_1,\ldots,\alpha_m)$ 的第 $i+1$ 个分量被发送给证明者 P，其他的分量都留作对其他副本的模拟。当 P 用消息 β 回答时，机器 V^*（通过提供给程序 P 相应的辅助输入和相应的输入消息序列）为 P 的其他拷贝计算答案。因此，V^* 可以分辨 P 使用证据 w_{i+1}^1 的情况和 P 使用 w_{i+1}^2 的情况。

这个证明可以很容易地扩展到多个证明系统异步方式同时执行的情况（即，顺序执行和并行执行是两个特殊情况）[①]。这个证明也可以扩展到强证据不可分辨性，因此我们得到引理 4.6.7。

引理 4.6.7（强证据不可分辨性的并行合成）：令 $L\in\mathcal{NP}$，R_L，(P,V)，Q，R_L^Q 和 P_Q 同引理 4.6.6 中的定义一样，如果 P（对于 R_L）是强证据不可分辨的，那么 P_Q（对于 R_L^Q）也是强证据不可分辨的。

4.6.3 构造

在本节中，我们给出证据不可分辨证明系统和证据隐藏证明系统的构造。

4.6.3.1 证据不可分辨证明的构造

使用并行合成引理（引理 4.6.7）并注意到零知识证明是（强）证据不可分辨的，我们可以得出定理 4.6.8。

定理 4.6.8：假设存在（非均匀）单向函数，那么 $\mathcal{NP}$ 中每种语言都有错误概率可忽略的常数轮数（强）证据不可分辨的证明系统。实际上，错误概率可以以指数减小。

我们指出，对于零知识证明系统没有这样的结果。也就是说，已知的 $\mathcal{NP}$ 证明系统

- 不是常数轮数的（例如，构造 4.4.9），或者
- 具有不可忽略的错误概率（例如，构造 4.4.7），或者

① 也就是说，证明系统的多项式次实例的执行可以任意交叉（由对手决定）。参见 4.12.2 小节中的进一步阅读建议。

- 需要更强的困难性假设(参见 4.9.1 小节),或者
- 仅仅是计算合理的(参见 4.9.2 小节)。

4.6.3.2 证据隐藏证明的构造

证据不可分辨的证明系统并不一定就是证据隐藏的。例如,任何具有惟一证据的语言都具有一个能产生惟一证据的证明系统(因此可能不是证据隐藏的),然而,这个证明系统显然是证据独立的。另一方面,对于某些关系而言,假设证明者是概率多项式时间的,那么证据不可分辨性就意味着证据隐藏性。例如:

命题 4.6.9:令$\{(f_i^0,f_i^1):i\in\overline{I}\}$是(非均匀)无爪(*claw-free*)函数的集合,且

$$R \stackrel{\text{def}}{=} \{(x,w):w=(\sigma,r)\wedge x=(i,x')\wedge x'=f_i^\sigma(r)\}$$

假定 P 对于 R 是证据不可分辨的概率多项式时间交互机,那么通过令 $i=I(1^n)$和 $x'=f_i^1(D(0,i))$产生的分布下,P 对于 R 也是证据隐藏的,其中 I 和 D 的定义与定义 2.4.6 中的相同。

非均匀无爪函数的集合是指非均匀电路族$\{C_n\}$,$\{C_n\}$中的函数除了以可以忽略的概率外不能在输入分布 $I(1^n)$上构造爪子。我们指出,前述的命题与和 P 之间交互的目的无关(例如,P 是否是语言中的成员、证据知识等)。

证明思路:命题通过反证法来证明。假设某个概率多项式时间交互机 V^* 在与 P 交互之后找到了证据。根据 P 的证据不可分辨性,不管 P 使用的证据是$(0,\cdot)$的形式还是$(1,\cdot)$的形式,V^* 都能很好地执行。用算法 D 将程序 V^* 和 P 结合起来,我们可以得到一个爪子形成算法(因此产生矛盾)。特别地,对于输入 $i\in\overline{I}$,这个爪子形成算法均匀选择 $\sigma\in\{0,1\}$,随机产生 $r=D(\sigma,i)$,计算 $x=(i,f_i^\sigma(r))$,并且模拟 V^* 和 P 在公共输入为 x、P 的辅助输入为(σ,r)时的交互。根据 P 的证据不可分辨性,V^* 的输出与 σ 的值是计算上独立的。因此,如果对于公共输入 x,机器 V^* 输出一个证据 $w\in R(x)$,那么以大约$\frac{1}{2}$的概率我们有 $w=(1-\sigma,r')$,这样就产生了一个爪子(因为 $f_i^\sigma(r)=f_i^{1-\sigma}(r')$)。最后,注意到我们需要分析爪子形成算法在输入分布 $I(1^n)$上的性能,并且在这种情况下,(P,V^*)模拟中的公共输入与命题中的假设是同分布的。

进一步,每个 **NP** 关系可以"稍作改动",使得对于改动后的关系,证据不可分辨性就意味着证据隐藏性。给定一个关系 R,修改的关系(记为 R_2)可以定义为

$$R_2 \stackrel{\text{def}}{=} \{((x_1,x_2),w):|x_1|=|x_2|\wedge \exists i \text{ s.t. } (x_i,w)\in R\} \tag{4.5}$$

即,w 是在 R_2 下实例(x_1,x_2)的证据,当且仅当 w 是一个在 R 下 x_1 或者 x_2 的证据。

命题 4.6.10:令 R 和R_2 同前面定义的一样,令 P 是一个对于 R_2 证据不可分辨的概率多项式时间交互机。那么 P 对于 R_2 在每个困难实例对的分布下都是证据隐藏的,其中的困难实例对是由有效算法随机选择 R 中的对而导出的。也就是说:

令 S 是一个对于输入 1^n、输出$(x,w)\in R$ 的概率多项式时间算法,并且$|x|=n$;X_n 代表输出 $S(1^n)$中的第一个元素导出的分布。假设$\{X_n\}_{n\in\mathbb{N}}$是 R 的困难实例集,那么 P 在总体$\{(X_n^{(1)},X_n^{(2)})\}_{n\in\mathbb{N}}$下是证据隐藏的,其中 $X_n^{(1)}$ 和 $X_n^{(2)}$ 表示 X_n 的两个相互独立的副本。

证明思路:令 S 和$\{X_n\}_{n\in\mathbb{N}}$同假设中定义的一样。假设某个交互机 V^* 在与 P 交互以后,以不可忽略的概率(在前述的分布下)找到了证据。根据 P 的证据不可分辨性,不管 P 使用的证据 w 对于公共输入(x_1,x_2),是满足$(x_1,w)\in R$ 还是满足$(x_2,w)\in R$,V^* 都会同样很好地

执行。用算法 S 将程序 V^* 和 P 结合起来，我们可以导出一个算法，记作 F^*，这个算法可以为 R 找到证据（在分布 X_n 下）：对于输入 $x\in L$，算法 F^* 会随机产生 $(x',w')=S(1^{|x|})$，并以 $\frac{1}{2}$ 的概率令 $\bar{x}=(x,x')$，其他的情况令 $\bar{x}=(x',x)$。算法 F^* 模拟 V^* 和 P 在公共输入为 $\bar{x}$、P 的辅助输入为 w' 时的交互，并且当 V^* 输出证据 w 时，算法 F^* 检查 $(x,w)\in R$ 是否成立。根据 P 的证据不可分辨性，验证者不能分辨 P 将证据应用于 $\bar{x}$ 的第一个元素的情况和应用于 $\bar{x}$ 的第二个元素的情况。根据 F^* 的构造，如果 F^* 的输入与 X_n 是同分布的，那么证明系统可以对公共输入为 $(X_n^{(1)},X_n^{(2)})$ 时进行模拟，其中 $X_n^{(1)}$ 和 $X_n^{(2)}$ 表示 X_n 的两个相互独立的副本。因此，根据前面的假设，V^* 以不可忽略的概率（在 x 和随机选择的 F^* 的分布上取）为 x 找到了证据。这样就得到 $\{X_n\}_{n\in\mathbb{N}}$ 对于 R 不是困难的。

4.6.4　应用

本节中提出的概念的应用散布在本书的各个章节中。特别地，强证据不可分辨证明系统在 $\mathcal{NP}$ 常数轮数零知识理论的构造中有应用（见 4.9.2 节），证据独立性证明系统在非同构图零知识证明中有应用（见 4.7.4.3 节），证据隐藏证明系统在基于因式分解的有效身份认证方案中有应用（见 4.7.5 节）。

4.7*　知识证明

本节介绍“知识证明”的概念。不严格地说，这个概念就是指这样的证明，在这个证明中证明者宣称知道某个对象的“知识”（例如，一个图的三着色），而不是仅知道知识的存在性（例如，图的三着色的存在性，这意味着图在语言 $G3C$ 中）。但是“机器知道某些事情”的意思是什么呢？事实上，本节的主要目的就是回答这个问题。在开始之前，我们首先指出“知识证明”，特别是零知识的“知识证明”在密码方案和密码协议的设计方面有许多应用。在 4.7.4 小节中讨论其中的一些应用。4.7.5 小节中讨论我们最感兴趣的认证方案方面的应用。最后，在 4.7.6 小节中我们介绍强知识证明的概念。

4.7.1　定义

4.7.1.1　动机讨论

“机器知道某些事情”的意思指什么？任何标准的词典对动词“知道”都有着多种解释，大多数的意思都是与“认识，了解”有关，这个概念当然不适用于我们所讨论的环境。我们必须给动词“知道”找一个行为化的解释。事实上，将知识同做事情的能力，至少是记录人们所知道的事情的能力联系在一起是合理的。因此，如果一个机器可以输出字符串 α，我们就应该说这个机器知道 α。但是这看起来像是彻头彻尾的废话：机器有一个明确定义的输出，该输出要么是 α，要么不是 α。“机器知道某些事情”究竟是什么意思呢？不严格地说，就是可以很容易地修改机器，使得无论什么样的要求都能完成。说得更准确些，就是存在一个有效机器，通过使用

原始机器作为黑盒子,可以按任何要求进行输出。

关于定义"机器的知识"我们已经说了这么多,然而无论机器知道什么还是不知道什么都是机器"自己的事情"。感兴趣且可作为外部参考的是,在与机器进行交互后可以从机器获得多少知识这个问题。因此,我们对知识证明(而不仅仅是知识)感兴趣。

为了简化,我们考虑一个具体问题:机器如何证明它知道一个图的三着色?一个显而易见的方法是简单地将这个图的三着色发送给验证者。然而,我们指出,应用构造 4.4.7(即 G3C 零知识证明系统)足够多次可以产生另一种替代的方案来证明关于图的三着色的知识。

不严格地讲,如果验证者被机器 P 说服并接受图 G 的概率与使用机器 P 作为黑盒时,获取 G 的三着色的难度成反比,我们就说交互机 V 构成了一个三着色知识的验证者。也就是说,三着色的获取由一个叫做提取器的预测机来完成,该提取器可以访问用来约束 P 的行为(即,它发送的消息与它可能接收到的特定消息相对应)的函数。我们要求,在输入为 G 和可访问用来约束 P 的消息的预测机的条件下,提取器的(期望)运行时间与 V 信任 P 并接受 G 的概率是成反比关系的(通过一个以 $|G|$ 为变量的多项式因式)。在 V 总是信任 P 而接受 G 时,提取器将会以期望多项式时间运行。当 V 信任 P 而接受 G 的概率较大时,这个命题仍然成立(我们强调,后面这种特殊情况对于满意的定义是不够的;参考 4.7.1.4 节中的高级注释)。

4.7.1.2 技术准备

令 $R\subseteq\{0,1\}^*\times\{0,1\}^*$ 是一个二元关系,那么 $R(x)\stackrel{\text{def}}{=}\{s:(x,s)\in R\}$,$L_R\stackrel{\text{def}}{=}\{x:\exists s\ \text{s.t.}\ (x,s)\in R\}$。如果 $(x,s)\in R$,那么我们称 s 是 x 的一个解。如果存在一个多项式 P,满足对于所有 $(x,s)\in R$,都有 $|s|\leq p(|x|)$,我们就说 R 是多项式限界的。如果 R 是多项式限界的,并且存在一个决定 R 中成员的多项式时间算法(实际上,有 $L_R\in\mathcal{NP}$),那么我们就说 R 是 $\mathcal{NP}$ 关系。在后面的讨论中,我们把研究范围限定在多项式限界关系上。事实上,本书中提到的所有应用都指的是 $\mathcal{NP}$ 关系。

我们希望能够以一种均匀的方式去考虑所有的潜在证明者,而不基于它们的运行时间、内部结构等等来区别它们。然而我们注意到,这些交互机能够接收一些辅助输入,这些辅助输入可以使交互机"知道"或者证明更多的东西。同样,它们可能幸运地选中一个最好的随机输入,从而比其他的交互机更能干。因此,关于证明者知识的描述不但同证明者的程序有关,而且同它接收的特定辅助输入和随机输入也有关。因此,我们固定交互机,也要固定这个交互机的所有输入(即公共输入、辅助输入和随机输入)。对于这样一种证明者+输入的模板,我们既要考虑和这个模板交互的(验证者)接受概率,又要考虑模板作为"知识提取器"的预言机的使用。这导致了如下定义。

定义 4.7.1(消息说明函数):记 $P_{x,y,r}(\overline{m})$ 为对于公共输入 x、辅助输入 y 和随机输入 r,在收到消息 $\overline{m}$ 后机器 P 所发送的消息。函数 $P_{x,y,r}$ 叫做机器 P 的具有公共输入 x、辅助输入 y 和随机输入 r 的**消息说明函数**。

可以访问函数 $P_{x,y,r}$ 的预言机能够表示具有公共输入 x、辅助输入 y 和随机输入 r 的机器 P 的知识。这个预言机叫做知识提取器,将尽量寻找 x 的一个解(即寻找 $s\in R(x)$)。要求提取器的运行时间与(与具有公共输入 x、辅助输入 y 和随机输入 r 的证明者 P 交互的验证者的)接受概率成反比关系。

4.7.1.3 知识验证者

既然所有机器都已经准备好了，我们就给出知识证明系统的定义。事实上，该定义是一个推广（是后面应用所需要的），在这个定义中，我们引入一个由函数 κ 表示的错误参数。在第一次阅读本节时，读者可以将函数 κ 看作等于 0。

定义 4.7.2（知识证明系统）：令 R 为一个二元关系，且令 $\kappa:\mathbb{N}\to[0,1]$。我们说交互函数 V 是关于关系 R 具有知识错误 κ 的一个知识验证者，如果下面的两个条件成立：

- 非平凡性：存在一个交互机 P，使得对于每对 $(x,y)\in R$，所有可能的 V 与具有公共输入 x 和辅助输入 y 的 P 的交互都是可接受的。
- （具有错误 κ 的）有效性：存在一个多项式 $q(\cdot)$ 和一个概率预言机 K，使得对于每个交互函数 P、每个 $x\in L_R$ 以及每个 $y,r\in\{0,1\}^*$，机器 K 都满足如下条件：

记 $p(x,y,r)$ 为在输入 x 下，交互机 V 同证明者（$P_{x,y,r}$）交互时 V 接受的概率。如果 $p(x,y,r)>\kappa(|x|)$，那么在输入 x 以及可访问预言机 $P_{x,y,r}$ 下，机器 K 以

$$\frac{q(|x|)}{p(x,y,r)\quad \kappa(|x|)}$$

限定的界限为上界的期望步骤的数目输出一个解 $s\in R(x)$。

我们称预言机 K 为通用知识提取器。

当 $\kappa(\cdot)$ 等于 0 时，我们简单地说 V 是一个关于关系 R 的知识验证者。满足 V 是关于关系 R 的知识验证者，P 是（关于 V 和 R）满足非平凡性条件的机器的交互机对 (P,V) 称为关于关系 R 的知识证明系统。

有效性条件的一个可选的公式化描述如下。假设存在概率预言机 K（如前所述）。然而，可选的公式化描述并不是要求 $K^{P_{x,y,r}}(x)$ 在与 $p(x,y,r)-\kappa(|x|)$ 反向相关的期望时间内总是输出一个解，而是要求 $K^{P_{x,y,r}}(x)$ 在期望多项式时间内，以至少 $p(x,y,r)-\kappa(|x|)$ 的概率输出一个解。实际上，我们还能通过要求以至少 $(p(x,y,r)-\kappa(|x|))/\mathrm{poly}(|x|)$ 的概率输出解来弱化这个可选的公式化描述。

定义 4.7.3（具有差错 κ 的有效性的可选公式化描述）：令 V，$P_{x,y,r}$（其中 $x\in L_R$）以及 $p(x,y,r)$ 同定义 4.7.2 中定义的一样。我们说 V 满足**具有差错 K 的可选有效性条件**，如果存在概率预言机 K 和一个正多项式 q，满足在输入 x 以及可访问预言机 $P_{x,y,r}$ 的情况下，机器 K 在期望多项式时间内运行并且以至少 $(p(x,y,r)-\kappa(|x|))/q(|x|)$ 的概率输出一个解 $s\in R(x)$。这两种有效性的公式化描述在 $\mathcal{NP}$ 关系的情况下是等价的。这种等价性基于这样的思想：在当前的环境中，成功概率和期望运行时间可以相互转化。

命题 4.7.4：令 R 为 $\mathcal{NP}$ 关系，V 为交互机。对于关系 R，机器 V 满足定义 4.7.2 中的（具有误差 κ）有效性条件当且仅当 V 满足定义 4.7.3 中的（具有误差 κ）可选有效性条件。

略证 假定 V 满足可选的公式化描述（具有误差 κ），令 K 为适合的提取器，q 为适合的多项式。因为已知 R 为 $\mathcal{NP}$ 关系，所以当调用 K 时，我们能够决定 K 是否成功。这样，我们就可以重复调用 K 直到 K 成功。如果 K 以 $s(x,y,r)\geq(p(x,y,r)-\kappa(|x|))/q(|x|)$ 的概率成功，那么期望调用次数为 $1/s(x,y,r)$，这与定义 4.7.2 中要求的一样。

假定 V 满足定义 4.7.2 的有效性要求（具有误差 κ），令 K 为适合的提取器，q 为适合的多项式（满足 K 以期望时间 $q(|x|)/(p(x,y,r)-\kappa(|x|))$ 运行）。令 p 为限定对于关系 R 的解

的长度的多项式(即$(x,s)\in R$,意味着$|s|\leq p(|x|)$)。然后我们连续执行$p(|x|)$轮:在第i轮中,我们模拟$K^{P_{x,y,r}}(x)$在$2^{i-1}\cdot q(|x|)$限定的时间内的运算。如果当前轮产生了一个解,那么我们停止运行并输出这个解,否则我们以$\frac{1}{2}$的概率继续执行下一轮(我们以特殊的失败符号来停止的概率也为$\frac{1}{2}$)。如果直到最后一轮都未得到一个解,那么我们通过穷举法搜索直到找到一个解(使用的时间为$2^{p(|x|)}\cdot \mathrm{poly}(|x|)$)。注意,第$i$轮得到执行的概率至多为$2^{-(i-1)}$,因此我们的期望运行时间至多为

$$\sum_{i=1}^{p(|x|)} 2^{-(i-1)}\cdot(2^{i-1}\cdot q(|x|))+2^{-p(|x|)}\cdot(2^{p(|x|)}\cdot \mathrm{poly}(|x|))$$
$$=4\cdot p(|x|)\cdot q(|x|)+\mathrm{poly}(|x|)$$

为了估计这个新的提取器的成功概率,注意到$K^{P_{x,y,r}}(x)$的运行时间是期望运行时间两倍(即$q(|x|)/(p(x,y,r)-\kappa(|x|))$的两倍)以上的概率不到$\frac{1}{2}$。还注意到在第$i\stackrel{\mathrm{def}}{=}-\log_2((p(x,y,r)-\kappa(|x|))$轮中,我们可模拟$2q(|x|)/(p(x,y,r)-\kappa(|x|))$步。这样,我们在前$i$轮中的某一轮提取一个解的概率至少为$\frac{1}{2}\cdot 2^{-(i-1)}=p(x,y,r)-\kappa(|x|)$,这与可选公式化描述中所要求的一样。

注释 命题 4.7.4 的证明实际上确立了:定义 4.7.2 的公式化描述隐含了定义 4.7.3 的公式化描述(这里$q=1$)。这样,定义 4.7.3 的公式化描述$q=1$与它的通用形式(即具有任意多项式q)是等价的。我们将在命题 4.7.5 和 4.7.6 的证明中使用这一结果。

4.7.1.4 讨论

考虑到命题 4.7.4,我们可以随意使用有效性公式化描述中的任何一个。当分析作为子协议的知识证明的影响时,定义 4.7.2 显然要方便一些,然而当证明一个给定的系统是知识证明时,定义 4.7.3 更为方便。我们指出,当R不是 **NP** 关系时,命题 4.7.4 的许多变种依然成立(参见习题 29)。

一点感想 知识证明的概念(和用来形式化知识证明的知识提取器的概念)同模拟规范有关。这一关系在一些应用中是很明显的,在这些应用中,知识验证者在相信知识证明者知道 $\boldsymbol{A}$ 之后采取一些动作 $\boldsymbol{K}$,这里动作 $\boldsymbol{A}$ 满足,如果知识证明者确实知道 $\boldsymbol{K}$,那么 $\boldsymbol{A}$ 对知识验证者是无害的。根据模拟规范,我们的定义指出,如果知识验证者在相信知识证明者知道 $\boldsymbol{K}$ 之后采取一些动作 $\boldsymbol{A}$,则对知识验证者无害,因为从某种意义上说,我们能够模拟知识证明者确实知道 $\boldsymbol{K}$ 的情况。实际上,通过使用知识提取器,我们能够模拟证明者关于整个交互的观点(即证明过程和随后由信任证明者的验证者所采取的行动):在证明者失败的情况下,将不再采取动作 $\boldsymbol{A}$,因此很容易模拟整个交互。如果证明者成功地获取了验证者的信任,那么我们可以提取相关知识 $\boldsymbol{K}$,并得到这样的情况:动作 $\boldsymbol{A}$ 是无害的(即 $\boldsymbol{A}$ 可基于 $\boldsymbol{K}$ 来模拟)。

关于合理性 在前述的定义中,我们没有对关于R的知识验证者在L_R以外的输入下被调用时所发生的情况提出要求。自然的要求是在输入$x\notin L_R$时,验证者接受的概率至多为$\kappa(|x|)$。这个结果在许多自然的情况下都成立,但是在命题 4.7.6 的结论中不成立。参考 4.7.3~4.7.6 小节中进一步的讨论。

进一步的探讨 有效性的两个公式化描述的一个关键特征是它们以一种"统一"的方式来处理$p(x,y,r)$所有可能的值。这对于大多数知识证明用作子协议(而不是最终协议)的应用(例

如,参考 4.7.4 小节)都是关键的。典型地,在前述应用中(即知识证明用作子协议的应用中),知识误差函数要求是可忽略不计的(甚至为 0)。在这些情况下,我们需要处理不能被忽略的 $p(x,y,r)$ 所有可能值,但是我们并不知道 $p(x,y,r)$ 的先验值。必须注意的是,$p(x,y,r)$ 是不可忽略的(作为 $|x|$ 的函数),并不意味着 $p(x,y,r)$ 是容易引起注意的(作为 $|x|$ 的函数)[①]。

4.7.2 减少知识误差

知识误差能够通过证明系统的顺序重复来减少。特别地,误差将会以重复次数的指数级来减少。

命题 4.7.5:令 R 为多项式限定的关系,$t:\mathbb{N}\to\mathbb{N}$ 是多项式限定的函数。如果 (P,V) 是一个关于关系 R、具有知识误差的知识证明系统,那么对于公共输入 x,由顺序重复 $t(|x|)$ 次 (P,V) 所得到的证明系统是一个关于关系 R、知识误差为 $\kappa'(n)\stackrel{\text{def}}{=}\kappa(n)^{t(n)}$ 的知识证明系统。

略证 令 (P',V') 表示将 (P,V) 顺序重复 t 次所得到的协议。为了分析 V' 的有效性,我们使用定义 4.7.3 的公式化描述。给定一个基本系统的提取器 K_1,我们如下构造复合系统 (P',V') 的提取器 K。对于输入 x,K 均匀选择 $i\in\{1,\ldots,t(|x|)\}$,模拟基本证明系统 (P,V) 的前 $i-1$ 轮叠代,并且以由前 i-1 次叠代副本所决定的剩余证明者为预言调用 K_1。也就是说,给定复合证明系统的证明策略的访问预言,我们首先用它来模拟 (P',V') 的前 $i-1$ 轮,这样产生一个副本 α。然后,我们定义基本证明系统的证明者策略,使得这个策略在 α 是前 $i-1$ 轮的副本的情况下,规定合成系统的证明者在第 i 轮的行为。将这个(基本系统)证明者策略用作预言,我们调用 K_1 来提取 x 的一个解。

剩下的工作就是来分析对于固定的 $x\in L_R$ 以及前面定义的 y 和 r,提取器 K 的成功概率。令 $t\stackrel{\text{def}}{=}t(|x|)$,$\kappa\stackrel{\text{def}}{=}\kappa(|x|)$,$\delta\stackrel{\text{def}}{=}p(x,y,r)-\kappa^t$(我们将在随后的符号中忽略这个固定的 (x,y,r))。我们的目的是说明 K 至少以 $\delta/\text{poly}(|x|)$ 的概率提取一个 x 的解。为了这个目的,我们用 a_{i-1} 来表示在复合证明系统的前 $i-1$ 轮中验证者每轮都接受的概率。对于前 $i-1$ 轮的每个可能的副本,我们用 $p'(\alpha)$ 来表示当 α 描述前 $i-1$ 轮副本时,在第 i 轮中验证者接受的概率。注意,如果 K 在模拟可产生副本 α 的前 $i-1$ 轮后试图提取一个解,则其成功概率至少为 $p'(\alpha)-\kappa$。令 c_i 为 $p'(\alpha)$ 的期望值,其中 α 在验证者接受的所有前 $i-1$ 轮副本中随机选取。那么 $a_i=a_{i-1}\cdot c_i$,并且 K 在模拟 $i-1$ 轮后可得到一个解的概率至少为 $a_{i-1}\cdot(c_i-\kappa)$。

结论:要么 $c_1-\kappa\geq\delta/t$,要么存在一个 $i\geq 2$ 使得 $a_{i-1}\cdot(c_i-\kappa)>\delta/t$。

在这两种情况中,我们都已经建立了一个足够低的 K 成功的概率下界;也就是说,K 的成功概率至少为 δ/t^2,这里 t 的一个特殊因子表示 K 会选择一个好的 i 的概率。

证明:注意,如果 $a_1=c_1<k+(\delta/t)$,那么一定存在一个 $i\geq 2$,满足 $a_{i-1}<\kappa^{i-1}+((i-1)\delta/t)$ 和 $a_i\geq\kappa^i+(i\delta/t)$,因为 $a_t=\kappa^t+\delta$。通过使用这个 i,我们有

$$
\begin{aligned}
a_{i-1}\cdot(c_i-\kappa) &= a_i-a_{i-1}\cdot\kappa \\
&> \left(\kappa^i+\frac{i\delta}{t}\right)-\left(\kappa^i+\frac{(i-1)\delta}{t}\right)
\end{aligned}
$$

---

① 回顾函数 $\mu:\mathbb{N}\to\mathbb{R}$ 是可以忽略的,如果对于每个正多项式 p 和所有足够大的 n,都有 $\mu(n)<1/p(n)$ 成立;而函数 $\upsilon:\mathbb{N}\to\mathbb{R}$ 是容易引起注意的,如果存在一个多项式 p 满足对于所有足够大的 n,都有 $\upsilon(n)>1/p(n)$ 成立。一个函数 $f:N\to R$ 可能既不是不可忽略的,也不是容易引起注意的;例如,考虑函数 f:$f(n)\stackrel{\text{def}}{=}2^{-n}$($n$ 为奇数),$f(n)\stackrel{\text{def}}{=}n^{-2}$($n$ 为偶数)。

$$= \frac{\delta}{t}$$

结论得证，同时命题也得证。

并行合成的情况怎样？ 和往常一样(参见 4.3.4 小节)，并行合成的效果要比顺序合成的效果更为复杂。因此，与命题 4.7.5 类似的结论在一般情况下不知道是否成立。同时，一些流行的知识的零知识证明的并行执行可以证明知识误差以轮数的指数级减少；参见问题 27。

消除微小误差：对于 **NP** 关系，无论知识误差什么时候小于通过随机猜测找到一个解的概率，我们都可以将知识误差设置为 0。

命题 4.7.6：令 R 为 **NP** 关系，$q(\cdot)$ 为一个多项式，使得 $(x,y)\in R$ 隐含有 $|y|\le q(|x|)$。假定 (P,V) 是关于关系 R、具有知识误差 $\kappa(n)\stackrel{\text{def}}{=}2^{-q(n)}$ 的知识证明系统，那么 (P,V) 是关系 R 的(具有零知识误差的)知识证明系统。

略证 我们还是使用定义 4.7.3 的公式化描述。给定一个满足假设条件的知识提取器 K，我们构造一个新的知识提取器，这个新的知识提取器首先调用 K，并且在 K 失败的情况下，它均匀选择一个长为 $q(|x|)$ 位的字符串，并且当且仅当该字符串是 x 的一个有效解时将其输出。令 $p(x,y,r)$ 同定义 4.7.2 和定义 4.7.3 中定义的一样，并且令 $s(x,y,r)\ge p(x,y,r)-\kappa(|x|)$ 表示 $K^{P_{x,y,r}}(x)$ 成功的概率。那么新知识提取器成功的概率至少为

$$s'(x,y,r)\stackrel{\text{def}}{=}s(x,y,r)+(1-s(x,y,r))\cdot 2^{-q(|x|)}$$

读者可以很容易地验证 $s'(x,y,r)\ge p(x,y,r)/2$(通过分开考虑 $p(x,y,r)\ge 2\cdot\kappa(|x|)$ 和 $p(x,y,r)\le 2\cdot\kappa(|x|)$ 这两种情况)，于是命题得证。

4.7.3 NP 知识的零知识证明

图同构零知识证明系统(即构造 4.3.8)和图三着色零知识证明系统(即构造 4.4.7)实际上都是相应语言的(具有一定知识误差的)知识证明。特别地，构造 4.3.8 是知识误差为 $\frac{1}{2}$ 的同构知识的证明，而构造 4.4.7 是知识误差为 $1-\frac{1}{|E|}$ 的图三着色知识的证明(当公共输入为 $G=(V,E)$ 时)；参见习题 26。通过循环执行每个构造足够多次，我们就能使知识误差以指数级减小(参见命题 4.7.5)。实际上，使用命题 4.7.6，我们可以得到零误差的知识证明。特别地，我们有：

定理 4.7.7：假定(非均匀)单向函数存在，那么每个 **NP** 关系都有关于知识证明的零知识系统。此外，非相应语言的输入被验证者接受的概率以指数级减小。

4.7.4 应用

我们简要回顾一下知识的(零知识)证明的应用。典型地，知识的零知识证明用来“相互透露”相同信息。假定 Alice 和 Bob 都声称知道某件事情(例如，公共输入图的三着色)，但他们对对方的声称相互置疑。在双方向使用知识的零知识证明事实上是使双方互相确信对方的声称的一个(概念上)简单的解决方案。

在开始描述这些应用之前，让我们来简要说明一下如何证明它们的安全性。典型地，知识的零知识证明一般用作子协议，在这个子协议中拒绝意味着验证方检测到了恶意欺骗。高层

协议的安全性证明通过模拟理论(simulation argument)来实现,这个模拟理论使用了知识提取器,但是只在验证方并未检测到恶意欺骗的情况下调用知识提取器。我们的知识证明的(有效性条件的)定义保证了,不论验证方接受的(先验未知)概率是多少,模拟器总会在期望多项式时间内运行。

在所有应用中,所采用的知识证明都具有可以忽略的合理性差错(即,非相应语言的输入被验证者接受的概率可忽略不计)。

4.7.4.1　有记忆的承诺方案

当使用承诺方案时,接收方确信,在承诺阶段之后发送方至多承诺一个值(也就是说,它以后只能"公布"这个值)。然而接收方并不确认发送方"知道"它承诺的是什么值。这样一种保证在多种情况下都可能是有用的,并且都可以通过使用知识证明来得到。欲知更多的细节,参考 4.9.2 小节。

4.7.4.2　防止选择明文攻击

防止对(公钥)加密方案的选择明文攻击的一种显而易见的方法是通过对明文进行知识的零知识证明来扩充密文。这样,选择明文攻击的效果(对攻击者来说)基本丧失。然而应该注意,所产生的加密方案采用的是在发送方和接收方之间(加密消息)的双向通信。(防止选择明文攻击的安全加密方案的定义及一些构造将在第 2 卷第 5 章中给出。)

4.7.4.3　*GNI* 零知识证明系统

在构造 4.2.8 中提出的非同构图(GNI)交互证明并不是零知识的(除非 $GNI \in \mathcal{BPP}$)。一个恶意验证者能够构造任意一个图 H,并通过发送 H 作为对证明者的一个质疑来了解 H 是否与第一个输入图同构。甚至可以强有力地驳斥构造 4.2.8 是辅助输入零知识的这个结论(例如,验证者能够检查它的辅助输入是否与作为公共输入的图中的某一个同构)。然而,我们注意到,如果验证者总是知道它的质疑的答案(就像它是诚实验证者一样),那么构造 4.2.8 "就应该能够成为零知识的"。我们通过让验证者向证明者证实它(即验证者)确实知道它的质疑图的答案(即,它知道一个合适输入图的同构),来修改构造 4.2.8 以得到 GNI 零知识证明,并且证明者只有在确信验证者可以正确回答它的质疑图时才对验证者的质疑进行答复。当然,验证者的知识证明也不应该泄露答案(否则证明者可以使用这些信息进行欺骗,这样不符合合理性要求)。如果验证者的知识证明是完备零知识的,那么它当然不会泄露答案。实际上,验证者的知识证明只要是证据独立的(如 4.6 节中定义的那样)就足够了。

4.7.5　身份证明(身份认证机制)

身份认证机制在所有用户之间不是很熟悉的大型分布式系统中是很有用的。在这样的系统中,允许用户向其他用户证明自己的身份。通过下面定义的身份认证机制可以实现这个目标。在下面,我们也将会看到身份认证机制与知识证明密切相关。我们提示一个人的身份可以与他做某件事情的能力特别是他证明某类知识的能力联系起来。

4.7.5.1 定义

不严格地说，身份认证机制由包含每个用户的记录的公开文件和身份认证协议组成。每条（公开）记录包括一个用户名（或者标识）和调用身份认证协议（后面将会谈到）时用到的辅助身份认证信息。该公开文件是由某受信任方建立和维护的，并且这个受信任方要担保这些记录的可靠性（即，每条记录是由它对应的用户提交的）。所有用户都可以随时访问公开文件。另外，受信任方也可以提供给每个用户它的公开记录的签名副本。现在假定 Alice 希望向 Bob 证明正在同他通信的确实是她本人。为此，Alice 用与她名字对应的（公开文件）记录作为参数来调用身份认证协议。Bob 经过验证，发现这个参数确实与 Alice 的公开记录相符，于是继续下一步操作，执行他在协议中的任务。要求 Alice 总是能使 Bob 相信（她确实是 Alice），而且其他任何人都不能欺骗 Bob 说她/他是 Alice。此外，即使接收到多项式个 Alice 发送的身份证明，Carol 也不能冒充 Alice。

Alice 使用随机算法来产生身份认证信息。很明显，如果身份认证信息将要使用，那么 Alice 必须秘密保存她要用来产生她的记录的随机硬币。而且，Alice 也必须在身份认证协议的使用过程中使用这些保存的硬币，但是使用时一定要保证以后不论什么人都不能冒充 Alice。

约定 在下面的定义中，我们采用（定义 4.2.10 中提出的）具有辅助输入交互机的公式化描述和符号约定。我们回顾一下，当 M 为交互机时，我们用 $M(y)$ 表示以固定的 y 作为 M 的辅助输入所得的交互机。在随后的定义中，n 为安全参数，在一般性稍有损失的情况下，我们假定用户名（或标识）是长度为 n 的字符串。如果 A 是概率算法，并且 $x,r\in\{0,1\}^*$，那么记 A 在输入 x 和随机硬币 r 的情况下的输出为 $A_r(x)$。

动机 下面定义中的算法 A 对应于产生身份认证信息的例程，而 (P,V) 对应于身份认证协议本身。交互机 B' 和 B'' 代表攻击者行为的两种组合（即，出于窃取机密的目的同用户交互，以及随后企图假扮用户）。在第一次阅读时，读者可忽略算法 B' 和（安全性条件中的）随机变量 T_n。然而，这样做会产生一个不能令人满意的安全性更弱的条件。

定义 4.7.8（身份认证机制）：身份认证机制由一个对 (I,Π) 组成，其中 I 是一个概率多项式时间算法，而 $\Pi=(P,V)$ 是一对概率多项式时间交互机，且满足如下条件：

- 生存性（viability）：对于每个 $n\in\mathbb{N}$、每个 $\alpha\in\{0,1\}^n$ 和每个 $s\in\{0,1\}^{\text{poly}(n)}$，都有
$$\mathsf{Pr}[\langle P(s),V\rangle(\alpha,I_s(\alpha))=1]=1$$
- 安全性（security）：对于每对概率多项式时间交互机 B' 和 B''、每个多项式 $p(\cdot)$、所有足够大的 $n\in\mathbb{N}$、每个 $\alpha\in\{0,1\}^n$ 和每个 z，都有
$$\mathsf{Pr}[\langle B''(z,T_n),V\rangle,(\alpha,Is_n(\alpha))=1]<\frac{1}{p(n)}$$

其中 S_n 是在 $\{0,1\}^{\text{poly}(n)}$ 上服从均匀分布的随机变量，T_n 表示在公共输入为 $(\alpha,Is_n(\alpha))$、交互次数为多项式次时，$B'(z)$ 和 $P(S_n)$ 交互后输出的随机变量。

算法 I 叫做信息生成算法，(P,V) **叫做身份认证协议**。

因此，为了使用身份认证机制，一个用户，比如说 Alice（她的身份用字符串 α 编码）应该首先均匀选择一个秘密字符串 s，计算 $i\stackrel{\text{def}}{=}I_s(\alpha)$，要求受信任的第三方将记录 (α,i) 写入公开文件中，并将字符串 s 放于安全的地方。生存性条件保证了 Alice 通过执行身份认证协议能够使 Bob 相信

她的身份：Alice 将存储的字符串 s 作为辅助输入调用程序 P，Bob 使用程序 V，并且将含有 α 的公开记录（这个公开记录已经被写入公开文件）作为公共输入。如果暂时不考虑算法 B' 和随机变量 T_n，那么安全性条件意味着：如果某一方只有 Alice 的公开记录和一些无关的辅助信息（通过辅助输入 z 来表示），那么这一方企图冒充 Alice 是不可能的。然而，这样的安全性条件可能在许多应用中是不够的，因为企图冒充 Alice 的用户可能首先要求 Alice 向他/她证明自己的身份。（完全）安全性条件保证了，即使 Alice 曾经多次向 Carol 证明过她的身份，Carol 冒充 Alice 也是不可能的。我们强调，如果 Carol 不能同时和 Alice 以及 Bob 交互，那么 Carol 就无法向 Bob 证明她是 Alice。在这个条件不成立时，什么都无法保证（事实上，Carol 可以通过将 Bob 的问题发给 Alice，将 Alice 的回答发给 Bob，从而很容易地冒充 Alice）。

4.7.5.2 身份验证机制和知识证明

验证某人身份的一个自然的方法就是要求他/她提供他/她应该知道的事实的知识证明。我们来考虑一个特定的（实际上是非常普遍的）例子。

构造 4.7.9（基于单向函数的身份认证机制）：令 f 为一个函数，在输入一个身份 $\alpha\in\{0,1\}^n$ 时，信息生成算法均匀选择一个字符串 $s\in\{0,1\}^n$ 并输出 $f(s)$（$(\alpha,f(s))$ 对是用户 α 的公开记录）。身份认证协议包括公开记录中第二元素逆元的知识证明。也就是说，为了证明它的身份，用户 α 证明它知道字符串 s 使得 $f(s)=r$，此处 (α,r) 是公开文件中的一条记录。（这里使用的知识证明允许有可忽略不计的知识误差。）

命题 4.7.10：如果 f 是一个单向函数，并且使用的知识证明是零知识的，那么构造 4.7.9 构成一个身份认证机制。

因此，如果单向函数存在，那么身份认证机制也存在。实际应用的身份认证机制可以基于特定的困难性假设来构造。例如，在假定因式分解困难的情况下，所谓的 Fiat-Shamir 身份认证机制（这种身份认证机制实际上是一个求模平方根的知识证明）成立。

构造 4.7.11（Fiat-Shamir 身份认证机制，基本版本）：在输入身份 $\alpha\in\{0,1\}^n$ 时，信息生成算法均匀选择一个由两个 n 位长的素数乘积构成的合数 N 和一个模 N 的余数 S，输出为 $(N, s^2 \bmod N)$（$(\alpha,(N,s^2 \bmod N))$ 对是用户 α 的公开记录）。身份认证协议包括相应的求模平方根的知识证明。也就是说，为了证明它的身份，用户 α 要证明它知道 $r \stackrel{\text{def}}{=} s^2 \bmod N$ 的一个平方根，其中 $(\alpha,(r,N))$ 是公开文件中的一条记录。（所使用的知识证明也允许可忽略不计的知识误差。）

求模平方根的知识证明与构造 4.3.8 中提出的图同构的证明系统是类似的。即为了证明知道 $r\equiv s^2 \pmod N$ 的平方根，证明者重复如下步骤足够多次。

构造 4.7.12（求模平方根知识的原子证明）：这个证明具有公共输入 (r,N)，其中指定的证明者具有辅助输入 s 使得 $r\equiv s^2 \pmod N$：

- 证明者随机选取一个模 N 的余数 g，并将 $h \stackrel{\text{def}}{=} g^2 \bmod N$ 发送给验证者。
- 验证者均匀选择 $\sigma\in\{0,1\}$，并将其发给证明者。
- 动机：在 $\sigma=0$ 时，验证者要求 $h \bmod N$ 的一个平方根，而 $\sigma=1$ 时，验证者要求 $h\cdot r \bmod N$ 的平方根。在后面的叙述中，不失一般性，我们假定 $\sigma\in\{0,1\}$。
- 证明者以 $a \stackrel{\text{def}}{=} g\cdot s^\sigma \bmod N$ 来应答。
- 当且仅当证明者所发送的消息 h 和 a 满足 $a^2\equiv h\cdot r^\sigma \bmod N$ 时验证者接受。

当构造 4.7.12 无论是串行还是并行重复 k 次以后，所生成的协议都是具有知识误差 2^{-k} 的一个求模平方根知识证明(参见习题 27)。当串行重复时，生成的协议是零知识的。然而，对于构造 4.7.11 中的使用要求来说，知识证明在相关分布下是证据隐藏的(参见定义 4.6.5)就足够了，即使当多项式个执行(以异步方式)同时发生时也是这样。因此，所生成的身份认证机制具有常数轮复杂度。我们指出，为了达到身份认证的目的，执行构造 4.7.12 超对数次就足够了。而且，较少次数重复执行得到的知识证明也可能是有用的：当应用构造 4.7.12 $k=O(\log n)$ 次，并使用构造 4.7.11 中生成的协议，我们就可以得到一个身份认证机制，在这个身份认证机制中，冒充用户成功的概率至多为 2^{-k}。

4.7.5.3 身份认证机制和能力证明

如前所述，字符串的知识证明(即，能够输出这个字符串的能力)是做某件事情的能力证明的一种特殊情况。于是身份验证机制也能够基于能力证明的一般概念来构建。我们不定义能力证明的概念，而是给出使用能力证明作为身份认证基础的两个"自然的"例子。

通过人们签名的能力来验证其身份是每日惯例，这个惯例可以引入到数字环境中。特别地，Alice 的公开记录由她的名字和验证密钥组成，这个验证密钥与预定的签名机制中的秘密签名密钥是对应的。身份认证协议包含 Alice 对验证者选择的一个随机消息进行签名。

另一种流行的身份认证方式通过用户正确回答个人问题的能力来验证。这种通用惯例的数字类比物如下。我们使用伪随机函数(参见 3.6 节)和(语言成员的)零知识证明。Alice 的公开记录包括她的名字和一个对于随机选择的伪随机函数的"承诺"(例如，或者通过对函数下标的字串承诺，或者通过一个由随机域元素和在该元素上的函数值构成的对)。身份认证协议包括 Alice 返回在验证者所选的随机位置上的函数值，并且提供如下情况的零知识证明：返回值确实与公开协议中出现的函数匹配。我们指出，这种数字化实现要比日常的惯例提供更好的安全性。在日常环境中，验证者知道所有可能的问题答案对的列表，并且相信验证者不会企图冒充用户。这里我们用"答案正确"的零知识证明代替了正确答案。

4.7.6 强知识证明

定义 4.7.2 和定义 4.7.3 本质上依赖于期望运行时间的概念。特别地，这两个定义都提到了期望运行时间的知识提取器。由于 4.3.1.6 节中所讨论的各种原因，只要可能，我们都会避开期望运行时间的概念。因此，下面我们考虑一个更为严格的定义，在这个定义中，要求知识提取器在严格多项式时间内运行，而不是在与接受概率成反比关系的期望多项式时间内运行(如定义 4.7.2 中定义的那样)。(我们在定义中假定接受 L_R 以外的实例的概率可忽略不计，这可通过将有效性条件的范围扩充到 x 在 L_R 之外来实现。)

4.7.6.1 定义

定义 4.7.13(强知识证明系统)：令 R 为二元关系。我们说交互函数 V 是一个**关于关系 R 的强知识验证者**，如果下面的两个条件成立：

- 非平凡性：同定义 4.7.2。
- 强有效性：存在一个可忽略函数 $\mu:\mathbb{N}\to[0,1]$ 和一个概率(严格)多项式时间预言机 K，

使得对于每个交互函数 P 和每个 $x,y,r\in\{0,1\}^*$，机器 K 满足如下条件：

令 $P(x,y,r)$ 和 $P_{x,y,r}$ 同定义 4.7.2 中定义的一样。如果 $p(x,y,r)>\mu(|x|)$，那么在输入为 x 且可访问预言机 $P_{x,y,r}$ 的情况下，机器 K 输出一个解 $s\in R(x)$ 的概率至少为 $1-\mu(|x|)$。

我们称预言机 K 为**强知识提取器**。

交互机对(P,V)叫做关系 R 的强知识证明系统，其中 V 是关系 R 的强知识验证者，而 P 是（关于 V 和 R）满足非平凡性条件的机器。

我们选择 μ（而不是另一个可忽略函数 μ'）作为提取器失败概率（在强有效性的要求下）的上界是无关紧要的。而且，对于 $\mathcal{NP}$ 关系，要求存在具有不可忽略成功概率的提取器等价于要求存在失败概率以指数级减小的提取器（也就是说，对于 $\mathcal{NP}$ 关系，连续应用提取器可减小失败概率）。这个强有效性要求比定义 4.7.2 中（具有误差 μ）的有效性要求从以下两个方面更强：

1. 无论 $p(x,y,r)$ 的值是多少，定义 4.7.13 中的提取器都在（严格）多项式时间内运行，而定义 4.7.2 中的提取器在期望时间 $\text{poly}(n)/(p(x,y,r)-\mu(|x|))$ 内运行。但是，要注意，定义 4.7.13 中的提取器的失败概率至多为 $\mu(|x|)$，而定义 4.7.2 中的提取器不会失败。

2. 强有效性要求意味着验证者接受 $x\notin L_R$ 的概率至多为 $\mu(|x|)$，而在定义 4.7.2 中没有这个要求。这个定义中的合理性条件是自然的，它与定义 4.7.2 不同，它允许非零（但可忽略）的错误概率。

4.7.6.2　一个例子：同构知识的强零知识(ZK)证明

连续重复图同构（零知识）证明系统（即构造 4.3.8）足够多次，可以产生一个强同构知识证明。关键一点是，基本证明系统（即构造 4.3.8）的每个应用可产生两种可能情况中的一种，具体产生哪种取决于验证者是要求观看第一个图的同构还是第二个图的同构。如果证明者在两种情况下都能正确应答，那么我们能够重现输入图之间的同构（通过将这两种情况下所提供的同构合成起来得到）。如果证明者在两种情况下都失败了，那么无论证明者从失败的那一刻起怎样去做，验证者都会拒绝。特别地，前述的讨论暗示了下面的强知识提取器的构造（这里我们要求重复基本证明系统 n 次，并且令 $\mu(n)=2^{-n}$）。

图同构的强知识提取器　在输入为(G_1,G_2)并且可访问证明者策略预言机 P^* 的情况下，我们从 $i=1$ 开始，重复 n 轮。开始时，T（当前副本）为空。

1. 从证明者策略得到中间图 G'（即 $G'=P^*(T)$）。

2. 从验证者两种可能的动作中提取证明者的答案。即，对于 $j=1,2$，令 $\psi_j\leftarrow P^*(T,j)$。如果 ψ_j 是 G_j 和 G' 的同构，我们就说 ψ_j 是正确的。

3. 如果两个 ψ_j 都不正确，那么 $\phi\leftarrow\psi_2^{-1}\psi_1$ 是 G_1 和 G_2 之间的同构。在这种情况下，我们输出 ϕ，然后停止。

4. 若 ψ_j 只对一个 j 正确（且 $i<n$），我们令 $T\leftarrow(T,j)$ 并且继续下一轮（即 $i\leftarrow i+1$）。否则停止运行，并且没有输出。

容易验证，如果提取器在第 i 轮（$i<n$）停止并且没有输出，那么验证者（在真实交互中）接受的概率为 0。类似地，如果提取器在第 n 轮停止并且没有输出，那么验证者（在真实交互中）接受的概率至多为 2^{-n}。因此，只要 $p((G_1,G_2),\cdot,\cdot)>2^{-n}$，提取器就可以成功恢复两个输入图之间的同构。

4.7.6.3 NP 关系知识的强零知识(ZK)证明

类似的理论也可应用于某些 NP 零知识证明系统。特别地，考虑下面关于汉密尔顿环路(HC)问题的基本(零知识)证明系统的 n 轮连续重复。我们考虑有向图(和有向汉密尔顿环路的存在性)。

构造 4.7.14(HC 基本证明系统)：

- 公共输入：一个有向图 $G=(V,E)$，其中 $n \stackrel{\text{def}}{=} |V|$。
- 证明者的辅助输入：G 中的一个有向汉密尔顿环路，$C \subset E$。
- 证明者的第一步(P1)：证明者选择一个顶点集 V 的随机置换 π，并对顶点置换后得到的图的邻接矩阵元素进行承诺。也就是说，证明者发送一个 $n \times n$ 的承诺矩阵，这个矩阵满足：如果$(i,j) \in E$，则令元素$(\pi(i),\pi(j))$承诺为 1，否则承诺为 0。
- 验证者的第一步(V1)：验证者均匀选择 $\sigma \in \{0,1\}$，并将其发送给证明者。
- 动机：$\sigma=0$ 意味着验证者要求检查承诺矩阵是否合法，而 $\sigma=1$ 意味着验证者要求出示置换图中的一个汉密尔顿环路。
- 证明者的第二步(P2)：如果 $\sigma=0$，那么证明者将 π 和所有承诺的公布(即原像)发送给验证者。否则证明者只将对元素$(\pi(i),\pi(j))$的承诺公布给验证者，其中$(i,j) \in C$。
- 验证者的第二步(V2)：若 $\sigma=0$，那么验证者检查公布的图是否通过 π 与 G 同构。否则，验证者只检查所有公布的值是否为 1，并且所有元素是否构成一个简单 n 环路(当然，在这两种情况中，验证者都要检查公布的值是否与承诺的值相符合)。验证者接受当且仅当相应条件成立。

我们得出结论，连续 n 次执行构造 4.7.14 所生成的协议是汉密尔顿环路知识的(零知识)强证明；参见习题 20 和 30。因为汉密尔顿环路问题是 NP 完全问题，所以我们得到 NP 中任何语言的(零知识)强证明系统。我们指出，已知的 NP 完全语言知识的(零知识)强证明在轮数复杂度方面开销都很大。而且，我们有如下定理。

定理 4.7.15：假定存在(非均匀)单向函数，那么每个 NP 关系都有一个知识强证明的零知识系统。

4.8* 计算合理性证明(参数)

在本节中，我们考虑交互证明系统概念的一个弱化版本。特别地，我们弱化交互证明系统的合理性条件。我们不是要求欺骗验证者去(以大于某个界限的概率)接受错误结论是不可能的，而是仅仅要求欺骗验证者是不可行的。我们称这样的协议为计算合理性证明系统(或讨论)。计算合理性证明系统的优点是，在某些合理的复杂性假设下，NP 中的所有语言都可以构建完备零知识计算合理性证明系统。注意到 NP 中所有语言都可能不存在完备零知识证明系统(见 4.5 节)。同时注意到，如果单向函数存在，那么 NP 中所有语言都存在计算零知识证明系统。因此，前面提到的肯定结果展示了 NP 零知识协议的合理性和零知识特性之间的某种折衷。(然而，我们指出，NP 完备零知识计算合理性证明是在比计算零知识证明更强的复杂性理论假设的基础上构建的。构造在更弱的假设下，特别是在只存在单向函数的假设下的

完备零知识计算合理性证明，实际上是一个很有趣的研究课题。)

我们指出，计算合理性证明系统看起来比一般的证明系统更有效。特别地，在一些似乎合理的复杂性假设下，***NP*** 中的任何语言都存在非常有效的计算合理性证明系统(即，只需要多项式对数次的通信和随机性)。除非 ***NP*** 包含在确定准多项式时间内(即，***NP*** $\subseteq$ Dtime(2^{polylog}))，否则相似的结果在普通的系统中不成立。

4.8.1 定义

计算合理性证明系统的定义可以从前面的讨论中很自然地得出。需要考虑的惟一问题是，仅将定义 4.2.4 中的合理性条件替换为计算合理性条件并不能产生自然的定义，因为(定义 4.2.4 中)完备性条件中的证明者的计算能力是没有限制的。因此，很自然就会将两个条件(完备性条件和合理性条件)中的证明者限制为一个有效的证明者。将"有效"解释为在给定辅助输入条件下的概率多项式时间，这一点是至关重要的(否则，只有 ***BPP*** 中的语言才有这样的证明系统)。因此，我们的出发点是定义 4.2.10(而不是定义 4.2.4)。

定义 4.8.1(计算合理性证明系统(理论))：一个交互机对(P,V)叫做**语言 L 的计算合理性证明系统**(或理论)，如果两个机器(关于辅助输入)都是多项式时间的，并且满足以下两个条件：

- 完备性：对于每个 $x\in L$，都存在一个字符串 y，使得对于每个字符串 z，都有

$$\Pr[\langle P(y),V(z)\rangle(x)=1]\geq \frac{2}{3}$$

- 计算合理性：对于每个多项式时间交互机 B、所有足够长的 $x\notin L$ 以及每个 y 和 z，都有

$$\Pr[\langle B(y),V(z)\rangle(x)=1]\leq \frac{1}{3}$$

和以前一样，通过足够多次地顺序重复执行协议，错误概率可以(从$\frac{1}{3}$)减小到 $2^{-\mathrm{poly}(|x|)}$；见习题 31。我们指出，并行重复在某些情况下可能不会降低(计算)合理性错误。

4.8.2 完备隐藏承诺方案

本节的主要目的是给出一个 ***NP*** 中每种语言的完备零知识理论的方法。这种方法本质上使用了具有完备(或者"信息理论")保密特性的承诺方案的概念。因此，我们先来说明这样的完备隐藏承诺方案。我们指出，这样的方案在其他环境中也可能有用(例如，在承诺的接受者不受计算限制的情况下，比如可以参考 4.9.1 小节中的例子)。

承诺方案(如定义 4.4.1 中定义的那样)与完备隐藏承诺方案(后面将要定义)之间的区别在于保密性要求和明确性要求范围的转换：在承诺方案中(见定义 4.4.1)，保密性要求是计算的(即，仅仅涉及概率多项式时间对手)，而明确性要求是信息理论上的(对对手的计算能力不做限制)。另一方面，在完备隐藏承诺方案中(见后面定义)，保密性要求是信息理论上的，而明确性要求是计算的(即，仅仅涉及概率多项式时间对手)。

术语注释 从现在开始，我们明确地提出我们谈及的承诺方案的"完备"特性。也就是说，定义 4.4.1 中的承诺方案是完备绑定的，而定义 4.8.2 中的承诺方案(后面将会给出)是完备隐藏的。因此，当我们谈及一个没有指定任何"完备"特性的承诺方案时，这个方案仅仅是计算隐藏和

计算绑定的。我们指出，不存在既是理想隐藏又是理想绑定的承诺方案(见习题 32)。

我们强调，前面刚刚建议的术语与 4.4 节中的说明不一致(在 4.4 节中，如定义 4.4.1 这样的方案只是称作“承诺方案”，而不强调“完备绑定”的额外特性)[①]。此外，刚刚建议的术语与本书中重要部分的术语也不一致，在这些重要部分中可以找到很多术语[②]。

4.8.2.1　定义

不严格地说，一个完备隐藏承诺方案就是一个有效的两阶段双方协议，通过这个协议发送者可以将它自己承诺为一个数值，并满足如下两个互相矛盾的要求：

1. (完备)保密性(或隐藏性)：在承诺阶段的结束，接收者不会得到任何有关发送者数值的信息。

2. 明确性(或者绑定)：发送者与接收者进行交互是不可行的，因此承诺阶段是成功中止的，然后发送者以两种不同的方式执行公布阶段，并导致接收者(作为合法的“公布”)接受两个不同的值。

使用与 4.4.1 小节中相似的约定，我们给出下面的定义。仍然用 S 和 R 分别来特指承诺发送者和接收者。

定义 4.8.2(完备隐藏比特承诺方案)：完备隐藏比特承诺方案是一对概率多项式时间交互机(S,R)，满足：

- 输入规范：公共输入是一个一元整数 n(作为安全参数)。发送者的私有输入是一个比特 v。
- 保密性(隐藏性)：对于每个与 S 交互的概率(不一定是多项式时间的)机 R^*，描述 R^* 在两种情况下输出的随机变量，即$\langle S(0),R^*\rangle(1^n)$和$\langle S(1),R^*\rangle(1^n)$是同分布的。
- 明确性(绑定性)：准备知识：为了简单起见，$v\in\{0,1\}$和 $n\in\mathbb{N}$ 在所有符号中都是默认成立的。固定任何概率多项式时间算法 F^* 和多项式 $p(\cdot)$。

1. 如定义 4.4.1 中那样，接收者对与发送者交互的观点记为$(r,\overline{m})$，包括接收者使用的随机硬币(即 r)和从发送者那里接收到的消息序列(即$\overline{m}$)。发送者对同一个交互的观点记为$(s,\widetilde{m})$，包括发送者使用的随机硬币(s)和从接收者那里收到的消息序列($\widetilde{m}$)。交互的合成观点是由这个交互的接收者观点和发送者观点构成的一个观点对。

2. 令$\sigma\in\{0,1\}$。我们说(交互的)合成观点$((r,\overline{m}),(s,\widetilde{m}))$(关于 F^* 和 $p(\cdot)$)具有一个**可行的 σ 公布**，如果当输入$(\overline{m},(s,\widetilde{m}),\sigma)$时，算法 F^* 以至少 $1/p(n)$的概率输出一个字符串s'，满足$\overline{m}$描述当 R 使用本地硬币 r 并与使用本地硬币 s'和输入$(\sigma,1^n)$的机器 S 交互时 R 接收到的消息。

(注意：我们强调，s'可以但不必等于 s。算法 F^* 一定要满足仅依赖于部分输入的关系(即接收者的观点$(r,\overline{m})$)。发送者的观点(即$(s,\widetilde{m})$)作为附加的帮助提供给算法 F^*。)

3. 我们说一个合成观点**关于 F^* 和 $p(\cdot)$是模糊的**，如果这个合成观点既有可行的 0 公

① 4.4 节的术语中省略这个额外特性的目的是为了简化基本描述。

② 例如，正如 4.4 节中那样，许多工作仅仅用术语“承诺方案”来提及如定义 4.4.1 中那样的方案，而许多工作又用术语“完备承诺方案”来提及如定义 4.8.2 中那样的方案。而且，在一些工作中，术语“承诺方案”就是指定义 4.8.2 中那样的方案。

布，也有可行的 1 公布（关于 F^* 和 $p(\cdot)$）。

明确性（绑定性）要求指出，对于除了可忽略的部分之外所有的接收者硬币投掷，发送者与接收者进行交互，不可能使得到的合成观点关于某些概率多项式时间算法 F^* 和某些确定多项式 $p(\cdot)$ 是不明确的。即，对于每个概率多项式时间交互机 S^*、概率多项式时间算法 F^*、正多项式 $p(\cdot)$ 和 $q(\cdot)$，以及所有足够大的 n，R 与 S^* 之间交互的合成观点在公共输入为 1^n 时，关于 F^* 和 $p(\cdot)$ 模糊的概率小于 $1/q(n)$。

在明确性要求的公式化描述中，S^* 描述（进行欺骗的）发送者在承诺阶段中的策略，而 F^* 描述发送者在公布阶段中的策略。因此，将发送者的（S^* 与 R 之间的）交互观点传给算法 F^* 是合法的（实际上也是必要的）。明确性要求指出，任何有效的策略 S^* 都不可能产生在后来可以被以两种方式（有效）公布并支持两个不同数值的交互合成观点。和以前一样，以可以忽略的概率发生的事件是可以忽略的。

可以考虑保密性条件放松的版本，在这个版本中，概率总体 $\{\langle S(0),R^*\rangle(1^n)\}_{n\in\mathbb{N}}$ 和 $\{\langle S(1),R^*\rangle(1^n)\}_{n\in\mathbb{N}}$ 不是同分布的，而是统计接近的。我们没有选择这样做的原因是当前已知的构造都可以达到更严格的条件。而且，使用完备隐藏承诺方案（4.8.3 小节）更弱的概念产生的是几乎完备零知识讨论，而不是完备零知识讨论。

与定义 4.4.1 中所述的一样，保密性要求明确指承诺阶段结束时的状态，而明确性要求隐含地假定公布阶段采用下面的规范形式：

1. 发送者将它的原始输入 v 和它在承诺阶段使用的随机硬币 s 发送给接收者。

2. 接收者要验证的是，v 和 s（以及 R 在承诺阶段中使用的硬币（即 r））确实产生了 R 在承诺阶段中接收的消息。验证过程是在多项式时间内完成的（通过执行程序 S 和 R）。

4.8.2.2 基于单向置换的构造

完备隐藏承诺方案可以用任何单向置换来构造。然而，已知的方案都包含线性（关于安全参数）轮数。因此，这些方案在本节中可以使用，而对于 4.9.1 小节中的构造就不能使用。

构造 4.8.3（完备隐藏比特承诺）：令 f 是一个置换，令 $b(x,y)$ 为 x 和 y 模 2 的内积（即，$b(x,y)=\sum_{i=1}^{n}x_iy_i \bmod 2$，其中 $x=x_1\cdots x_n\in\{0,1\}^n$，$y=y_1\cdots y_n\in\{0,1\}^n$）。

1. 承诺阶段（使用安全参数 n）：

(a)（本地计算）：接收者随机选择 $n-1$ 个线性无关的向量 $r^1,\ldots,r^{n-1}\in\{0,1\}^n$。发送者均匀选择 $s\in\{0,1\}^n$，并计算 $y=f(s)$。

(b)（迭代哈希）：交互双方运行 $n-1$ 轮。在第 i 轮中（$i=1,\ldots,n-1$），接收者将 r^i 发送给发送者，发送者计算并发送 $c^i \stackrel{\text{def}}{=} b(y,r^i)$。

(c)（“实际”承诺）：现在，等式系统 $\{b(y,r^i)=c^i:1\leq i\leq n-1\}$ 恰好有两个解（交互的双方都可以很容易地确定这两个解）。

- 如果 y 是（两个解中）按照字母顺序的第一个解，那么发送者就令 $\pi=1$，否则就令 $\pi=0$。
- 为了承诺为数值 $v\in\{0,1\}$，发送者发送 $c^n \stackrel{\text{def}}{=} \pi\oplus v$ 给接收者。

2. 规范公布阶段：在公布阶段中，发送者公布 v 和它在承诺阶段随机选择的字符串 s。如果如下两个条件成立，接收者就接受数值 v，其中 $((r^1,\ldots,r^{n-1}),(c^1,\ldots,c^n))$ 表示接收者对承诺阶段的观点：

- 对于所有 $1\leq i\leq n-1$，$b(f(s),r^i)=c^i$。

- 如果存在($y'<f(s)$(或者 $y'>f(s)$),使得对于所有的 $1\le i\le n-1$,都有 $b(y',r^i)=c^i$,那么一定满足 $v=c^n$(或者 $v=c^n\oplus 1$)。

也就是说,接收者解线性系统$\{b(y',r^i)=c^i\}_{i=1}^{n-1}$,得到解 $y^1<y^2$,使得对于 $j=1,2$ 和 $i=1,\ldots,n-1$,都有 $b(y^j,r^i)=c^i$。接下来,接收者检查 $f(s)\in\{y^1,y^2\}$是否成立(如果不成立,就立即拒绝),并且设置相应的 π(即,使得 $f(s)=y^\pi$ 成立)。当且仅当 $v\equiv c^n+\pi \pmod 2$时,验证者接受数值 v。

命题 4.8.4:假设 f 是一个单向置换,那么构造 4.8.3 中提出的协议就构成了一个完备隐藏比特承诺方案。

很容易就可以看出,构造 4.8.3 满足保密性条件。满足明确性要求的证明十分复杂,因此我们就忽略了。证明的直觉思想是叠代进行哈希,使得对解 y^1 和 y^2 求逆是不可行的(我们指出,如果用普通哈希来代替迭代哈希,这个推理就不成立了;见习题 33)。

4.8.2.3 基于(无爪)集合的构造

常数轮的完备隐藏承诺方案可以通过使用看起来更强的不可逆假设,即无爪集合(见 2.4.5 小节)来构造。这个假设隐含了单向函数的存在性,但是并不知道逆命题是否成立。然而,无爪集合可以在如整数分解的困难性和离散对数问题(DLP)等广泛相信的假设下建立。实际上,接下来要提到的完备隐藏承诺方案的构造,使用了具有附加特性的无爪集合;特别地,这个构造假设无爪集合的下标集(即算法 I 的范围)可以被有效地识别出来(在 ***BPP*** 中)。这样的集合在 DLP 难解的假设下存在(见 2.4.5 小节)。

构造 4.8.5(一个常数轮完备隐藏比特承诺):令(I,D,F)是一个概率多项式时间算法的三元组(I 是无爪集合$\{(f_i^0,f_i^1):i\in\bar{I}\}$的下标产生算法,$S$ 和 F 是相应的取样和评估算法)。

1. 承诺阶段:为了接收一比特(使用保密参数 n)的承诺,接收者随机产生 $i=I(1^n)$,并发送给发送者。为了承诺为数值 $v\in\{0,1\}$(在从接收者那里接收到消息 i 后),发送者检查 i 是否真的在 $I(1^n)$的范围中,如果在这个范围中,发送者就随机产生 $s=D(v,i)$,计算 $c=F(v,i,s)$,并将 c 发送给接收者(当 i 不在 $I(1^n)$的范围中时,发送者就中止协议,并宣布接收者正在进行欺骗行为)。

2.(几乎规范)公布阶段:在公布阶段中,发送者公布由它自己在承诺阶段产生的字符串 s 就足够了。如果 $F(v,i,s)=c$,接收者就接受数值 v,其中(i,c)是接收者对承诺阶段的(部分)观点。

命题 4.8.6:令(I,D,F)是具有概率多项式时间可识别下标集的无爪集合,那么构造 4.8.5 中给出的协议构成一个完备隐藏比特承诺方案。

略证 保密性要求可以直接从无爪集合(见定义 2.4.6)的特性 2 和发送者执行的测试 $i\in I(1^n)$得到。明确性要求从无爪集合(定义 2.4.6)的特性 3,运用标准可约性理论得到。(注意,$F(0,i,s_0)=F(1,i,s_1)$意味着(s_0,s_1)构成了置换对(f_i^0,f_i^1)的一个爪子。)

要求发送者检验下标 i 是否真正属于合法的下标集$\bar{I}$的基本原理是,只有满足 $i\in\bar{I}$的置换对(f_i^0,f_i^1)能够保证具有相同的值域分布。因此,实际上发送者没有必要检验 $i\in\bar{I}$是否成立,验证者只要检验(或者相信)置换对(f_i^0,f_i^1)满足同值域分布的要求就足够了。例如,考虑因数分解无爪集合(见 2.4.5 小节)。并不知道这个集合是否具有可有效识别的下标集。而且,在发送了一个下标 N 后,接收者可以以零知识的方式向发送者证明,置换对(f_N^0,f_N^1)满足同

值域分布的要求。实际上要证明的是，每个模 N 二次剩余的平方根的一半具有雅克比符号 1（相对于 N）。这个结论的（完备）零知识证明系统确实存在（不需要做任何假设）。事实上，使用证据独立证明系统就足够了，而且具有固定轮数的证据独立证明系统确实存在（同样不需要任何假设）。因此，因数分解无爪集合可以用来建立固定轮数的完备隐藏承诺方案，并且在 Blum 因数分解困难的假设下，这样的承诺方案也存在。

4.8.2.4 非均匀计算明确性

实际上，对于证明/理论系统的应用，不仅有接下来所要说的应用还有在 4.9.1 小节中的应用，我们都需要具有完备保密性和非均匀计算明确性的承诺方案（这个需要的原因与 4.4 节中 **NP** 零知识证明情况下的讨论是相似的）。非均匀计算明确性是指明确性条件对于（非均匀）多项式规模电路族也应该成立。我们强调，如果困难性假设关于非均匀多项式规模电路（例如，甚至用这样的电路单向置换也很难求逆无爪集合也很难有非均匀多项式尺寸的爪子形成电路）也成立，那么前面构造的完备承诺方案具有非均匀计算明确性。

为了避免术语带来的烦琐，当我们在前面提到的两个应用的描述中提到完备隐藏承诺方案时，我们省略属性“非均匀”。

4.8.2.5 具有后验保密性的承诺方案

我们继续讨论具有更松的保密要求的完备隐藏承诺方案。得到的方案不适用于本节的论题，然而，它在后面讨论的不同环境中是很有用的。弱化的优点是它允许我们使用任何无爪集合来构造这样的（固定轮数完备隐藏）承诺方案，因而，也就不再需要下标集是可有效识别的额外要求。

不严格地说，我们通过要求仅当接收者遵照它预定的程序（R）执行时满足保密性要求，来弱化完备隐藏承诺方案的保密性要求。这样看起来很奇怪，因为我们并不是真的要假设真正的接收者遵守预定的程序（而是对于接收者的任意行为都能提供保护）。问题在于真正的接收者可以在后面的阶段，甚至在公布阶段之后公布它在承诺阶段中使用的硬币投掷，并且通过这样做，接收者可以证明一个后验事实（至少在某种较弱的意义下），即它遵循了预定的程序。实际上，接收者证明的仅仅是它的行为方式与它的程序是一致的。

定义 4.8.7（具有完备后验保密性的承诺方案）：除了保密性要求由以下的后验保密性要求来取代之外，**具有理想后验保密性的比特承诺方案**的定义与定义 4.8.2 相同：对于每个字符串 $r\in\{0,1\}^{\text{poly}(n)}$，都有 $\langle S(0),R_r\rangle(1^n)$ 和 $\langle S(1),R_r\rangle(1^n)$ 是统计接近的，其中 R_r 表示当使用内部硬币投掷 r 时交互机器 R 的执行。

命题 4.8.8：令 (I,D,F) 是一个无爪集合。考虑构造 4.8.5 的修改版本，在这个修改版本中（承诺阶段）忽略了发送者对 i 是否在 $I(1^n)$ 范围内的检验。那么得到的结果协议就构成了一个具有完备后验保密性的比特承诺方案。

我们强调，与命题 4.8.6 不同，这里的无爪集合不需要具有有效识别下标集。因此，我们不得不忽略发送者的检验。然而，接收者后来可以简单地通过揭示它用来产生 i（使用算法 I）的随机硬币，来证明它在承诺阶段中发送的消息（即 i）确实是一个有效的下标。

略证 后验保密性要求可以从无爪集合的特性 2（以及 i 确实是一个有效下标这个事实，

因为 i 是通过调用 I 产生的)直接得到。明确性要求从命题 4.8.6 可以得到。

具有后验保密性的承诺方案的一个典型应用将在 4.9.1 小节中给出。在那里,承诺方案在一个交互证明中使用,在这个交互证明中,验证者扮演的是发送者的角色(证明者扮演的是接收者的角色)。如果验证者后验得知证明者进行了欺骗,那么验证者就会拒绝这个输入。因此,在这种情况下,验证者承诺的保密性可能被违背的事实,不会产生任何损害。

4.8.3　*NP* 完备零知识理论

在我们的环境中具有了一个完备隐藏承诺方案,我们就可以通过对(计算)零知识证明(对于 *NP*)的构造进行纯句法上的修改,构造 *NP* 完备零知识理论。我们知道在这些证明系统中(例如,关于图的三着色问题的构造 4.4.7),证明者使用完备绑定承诺方案将自己承诺为多个数值,然后证明者根据验证者的要求将这些数值中的一些公布出来。要做的所有事情就是,将验证者使用的完备绑定承诺方案替换为完备隐藏承诺方案。我们指出,得到的结果协议是原有语言的完备零知识讨论(即,计算合理性证明)。

命题 4.8.9:考虑对构造 4.4.7 的修改,在这个修改中将证明者使用的承诺方案替换为完备隐藏承诺方案,那么得到的结果协议是一个关于图的三着色的完备零知识弱理论。

弱理论指的是一个协议,在这个协议中,完备性条件和计算合理性条件之间的间隔是明显的。在我们讨论的情况中,验证者总是接受 $G3C$ 中的输入,而没有一个有效证明者可以以大于 $1-\frac{1}{|E|}$ 的不可忽略的概率来欺骗验证者接受不在 $G3C$ 中的图形 $G=(V,E)$。特别地,我们将会指出,没有一个有效证明者可以以大于 $1-\frac{1}{2\cdot|E|}$ 的概率来欺骗验证者接受不在 $G3C$ 中的图形 $G=(V,E)$。注意,通过(顺序)重复这个协议多项式次,(计算合理性)错误概率可以减小到可忽略的程度。

略证　我们先来证明得到的结果协议是 $G3C$ 完备零知识。我们使用与命题 4.4.8 的证明中相同的模拟器。然而,这里分析模拟器特性要更简单并会产生更强的结果,原因是在这里证明者的承诺是完备隐藏的,而在命题 4.4.8 的证明中仅仅是计算隐藏的。因此,这里的证明者承诺与承诺的数值之间是独立分布的,从而验证者的行为与这些数值是完全独立的。因此模拟器以 $\frac{2}{3}$ 的概率输出一个副本,同样,这个副本与真正的交互之间是独立分布的。这样就证明了完备零知识的特性。

与命题 4.4.8 的证明一样,完备性条件是显然的。接下来要证明协议满足(弱)可计算合理性要求。实际上,这是本证明中更细节的部分(与命题 4.4.8 的证明相反,命题 4.4.8 的合理性证明很容易)。原因是这里证明者承诺仅仅是计算绑定的,而在命题 4.4.8 的证明中是完备绑定的。因此,在这里我们使用可约性理论来说明对于不在 $G3C$ 中的输入,证明者以太高概率欺骗的能力,可以转换为一个与承诺方案的明确性相反的算法。下面是证明细节。

我们假设,存在一个(多项式时间)进行欺骗的证明者 P^* 和一个无限整数序列,使得对于这个序列中的每个整数 n,都存在图 $G_n=(V_n,E_n)\notin G3C$ 和一个字符串 y_n,满足 $P^*(y_n)$ 会使验证者以大于 $1-\frac{1}{2\cdot|E_n|}$ 的概率接受 G_n,令 $k\stackrel{\text{def}}{=}|V_n|$,令 $c_1,\ldots,c_k$ 是证明者在步骤 P_1 中发送的(对顶点颜色的)承诺序列。在下一步中,验证者发送一个均匀选择的边(属于 E_n),证明者必须通过公布这条边的两个端点的不同颜色来作答;否则,验证者就会拒绝接受。直接计算显

示，因为 G_n 是不可三着色的，就一定存在一个顶点，证明者对这个顶点至少公布了两种不同的颜色。因此，我们可以构造一个集成了 P^*、G_n 和 y_n 的多项式规模电路，这个电路违背了(非均匀)明确性条件。这样就得到与命题中的假设矛盾，从而完成了证明。

结合命题 4.8.4 和命题 4.8.9，我们可以得到如下推论。

推论 4.8.10：如果存在非均匀单向置换，那么 ***NP*** 中的每种语言都有完备零知识理论。

零知识证明和完备零知识理论：哪个更好？

命题 4.4.8 和命题 4.8.9 显示了合理性强度和零知识特性之间的一种折衷。命题 4.4.8 的协议提供了计算零知识和"完备"合理性，而命题 4.8.9 的协议给出了完备零知识和计算合理性。我们指出，这两种结果不是在相同的假设下得出的：只要单向函数存在，命题 4.4.8 的结论就是有效的，而命题 4.8.9 的结论需要(或许)更强的假设。然而，在假设两个协议都有效的情况下(即复杂性假设成立)，可能有人会问哪个更可取。答案取决于协议使用的环境(即应用)。特别地，需要考虑以下问题：

- 特定应用中合理性和零知识的相对重要性。如果两个特性中的一个具有明显的优越性，就应该做相应的选择。
- 应用中不同用户的计算资源。其中的一个用户可能会有更多的基本计算资源，因而应该倾向于要求他/她不可能进行欺骗，甚至要求在信息理论意义下也不能进行欺骗。
- 合理性要求仅与执行期间有关，而在许多应用中可能更关注长时间的零知识特性。如果情况确实如此，那么完备零知识理论就确实(比零知识证明)具有更明显的优点。

4.8.4 多项式对数效率的讨论

NP 零知识讨论的效率的大幅度提高，可以通过将认证树的想法与概率可检验证明(PCP)的结果结合起来得到。特别地，假设存在很强的无冲突哈希函数，那么仅使用多项式对数数量的通信和随机性，就可以构造 ***NP*** 中任何语言的计算合理(零知识)证明。在这个结论中有趣的一点是，仅考虑这样极其有效的讨论的存在性，而不管它们的零知识特性。因此，我们仅考虑与构造这样的理论有关的思想，而不涉及使这些理论成为零知识的那些问题(我们强调，接下来要提出的理论系统不是零知识的，除非 ***NP***⊆***BBP***)。

根据所谓的 PCP 定理，每个 ***NP*** 语言 L 可以简化为 $3SAT$，于是 L 的非成员被映射为 3CNF 公式，其中分配给它们的每个真值最多满足这些子句中的 $1-\epsilon$，其中 $\epsilon>0$ 是一个通用常数。我们用 f 表示这个约简。现在，为了证明 $x\in L$，只要证明公式 $f(x)$ 是可满足的就足够了。通过为 $f(x)$ 提供一个满足的分配就可以做到这一点。有趣的是，验证者不需要检验 $f(x)$ 的所有子句都满足给定的分配，而是均匀地选择多项式对数个子句，并检验分配对于这些子句是否满足。如果 $x\in L$(并且证明者为 $f(x)$ 提供了满足的分配)，那么验证者将总是接受。但是如果 $x\notin L$，那么将没有分配会满足子句中的 $1-\epsilon$，因此一个均匀选择的子句不会满足的概率至少为 ϵ。因此，检查超对数个子句就可以得到结果。

前面一段说明了随机复杂度可以变为多项式对数的，并且验证者仅需要检验多项式对数个随机选择的数值。特别地，证明者对公式 $f(x)$ 中变量的每个值都进行承诺，但是只会被要求公布其中一部分值。为了得到(总体的)多项式对数通信复杂性，我们使用一个特别的承诺方案，这个方案允许我们承诺为一个长度为 n 的字符串，以致子承诺阶段

采用多项式对数通信，并且这个字符串的单个比特都可以用多项式对数通信代价公布出来(并被验证为是正确的)。为了构造这样的承诺方案，我们使用无冲突哈希函数。这个函数将某个长度的字符串映射为那个长度的一半的字符串，并满足很“难”找到两个字符串会被这个函数映射为相同的像。(下面的描述有些不准确。我们需要的是一个哈希函数族，这个函数族中的函数满足在给定函数描述的情况下，没有小规模的非均匀电路可以构造出关于这个函数的冲突来。)

令 n 为输入字符串的长度，发送者希望将自己承诺为 n，令 k 是一个参数(之后将会被设置为 n 的多项式的对数)。用 H 表示一个将长度为 $2k$ 的字符串映射为长度为 k 的字符串的无冲突哈希函数。发送者将它的输入分割为 $m \stackrel{\text{def}}{=} \frac{n}{k}$ 个连续块，每块的长度为 k。接下来，发送者构造一个深度为 $\log_2 m$ 的二叉树，将这 m 个块放置在树中相应的树叶上。在每个内部节点上，发送者放置通过将函数 H 应用于该节点的孩子的内容而得到的哈希值。在承诺阶段中发送的惟一消息是根的内容(由发送者发送给接收者)。除非发送者可以构造在 H 下的冲突，否则通过这样做，发送者就可以将自己承诺为一个 n 比特长的字符串。当接收者想要得到这个字符串中特定的一位时，发送者就沿着从根部到相应的叶子的路径将每个节点的两个孩子的内容都公布给接收者。接收者检查提供给(沿着路径的)每个节点的数值和通过将 H 应用于该节点的两个孩子提供的数值而获得的数值是否匹配。

宣称 $x \in L$ 的协议包括：证明者使用前面的方案将自己承诺为一个 $f(x)$ 满足的分配，以及验证者通过要求证明者公布分配给一些子句中的变量的数值来检验这些子句。如果不可能找到特殊的一对 $\alpha, \beta \in \{0,1\}^{2k}$ 使得 $H(\alpha) = H(\beta)$，那么这个协议可以证明是计算合理的。特别地，我们需要假定在次指数时间内构造 H 的冲突是不可能的，即对于某个 $\delta > 0$，要以大于 2^{-k^δ} 的概率构造冲突必须需要至少 2^{k^δ} 时间。在这样的情况下，我们设置 $k = (\log n)^{1+\frac{1}{\delta}}$，并得到一个通信复杂度为 $O(\frac{\log n}{o(1)} \cdot (\log m) \cdot k) = \text{polylog}(n)$ 的计算合理性证明(通过恰当设置参数 k，冲突构造任务的更弱的下界可能会产生有意义的结果；比如说，对于任何固定的 $\epsilon > 0$，爪子不能在多项式时间内形成的标准假设允许我们令 $k = n^\epsilon$，并得到通信复杂度 $n^{\epsilon + o(1)}$)。我们强调，冲突总能在 2^{2k} 时间内形成，因此如果证明者不是计算资源限定的话，整个方法都会失效(因此，我们就不能用这种方法获得(完备合理)证明系统)。此外，可以说明只有 Dtime(2^{polylog})中的语言才有具有多项式对数的通信复杂度和随机复杂性的证明系统。

4.9* 常数轮零知识证明

在本节中，我们考虑 ***NP*** 中所有语言的具有可忽略错误概率常数轮零知识证明系统的构造问题。为了使接下来的讨论变得简单一些，我们定义证明系统是**轮有效的**，如果这个证明系统既是固定轮数的，又有可忽略的误差概率。我们强调，到目前为止所提出和讨论的 ***NP*** 零知识证明系统没有一个是轮有效的(即，它们或者是没有常数轮，或者是没有可忽略的误差概率)。

我们提出 ***NP*** 轮有效零知识证明的两种构造方法：

1. 基于常数轮完备隐藏承诺方案(如 4.8.2 小节中所定义的)构造轮有效零知识证明系

统。

2. 构造(轮有效零知识)计算合理证明系统(如 4.8 节中所定义的)而不是(轮有效零知识)证明系统。

第二种方法的优点在于,**NP** 轮有效零知识计算合理证明系统可通过使用任何单向函数来构造,但是并不清楚在同样的假定下,**NP** 轮有效零知识证明系统是否可以构造。特别地,我们知道怎样仅通过使用看起来更强的假设(例如,无爪置换的存在性)来构造常数轮完备隐藏承诺方案。

这两种方法的基本思想是相同的。我们先来简要介绍一下这个基本思想。注意到构造 4.4.7 中提出的图三着色基本零知识证明包括固定的轮数。然而,这个证明系统具有不可忽略的错误概率(实际上,错误概率非常接近于 1)。在 4.4 节中通过连续串行地应用证明系统足够多次来使误差概率降低到可忽略的水平。问题是,这使得证明系统的轮数不再是固定的。一种自然的建议是并行而不是串行重复执行这个基本证明。这种"解决方法"的问题在于不知道所生成的证明系统是否是零知识的。此外,正如命题 4.4.8 的证明中所做的那样,提出一个将所有可能的验证者作为黑盒子(参见 4.5.4 小节)的模拟器是不可能的。问题源于当并行运行构造 4.4.7 的多个副本的时候,恶意验证者能够根据在所有副本中(即步骤 P1)发送的承诺,在每个副本中选择将要检查的边(即步骤 V1)。验证者的这种行为使得与命题 4.4.8 的证明中提出的模拟器相似的模拟器无效。

克服这一困难的一种方法是"交换"步骤 P1 和 V1 的顺序,但是交换这些步骤的顺序会使证明者有了欺骗的可能(通过发送只有"查询边"正确着色的承诺进行欺骗)。因此,需要一个更精练的方法。首先,验证者对每个(构造 4.4.7 的)副本将自己承诺为一个边查询,然后验证者将自己承诺为每个副本中的着色,在这之后验证者公布它的查询,接下来的证明过程与前面的相同。验证者使用的承诺方案应该能够防止证明者预测出验证者承诺的边顺序。下面是这两种方法的不同点。

1. 第一种方法使用完备隐藏承诺方案。这种方法的问题在于,这样的(常数轮)方案只能在看起来比单向函数的存在性更强的假设下才存在。这样的方案在例如整数分解的困难性和离散对数问题的困难性的假设下确实存在。

2. 第二种方法限制了积极进行恶意欺骗的证明者的计算资源。因此,"对于"这些证明者(和承诺的接收者),使用具有计算安全性的承诺方案就足够了。我们指出,这种方法(对证明者的承诺)使用具有额外属性的承诺方案,而且这样的承诺方案可以使用任意单向函数来构造。

注意事项 在不严格的意义上(即,使用 4.3.1.6 节中定义的期望多项式时间模拟器),这两种方法都能生成零知识协议。并不清楚这些协议(或者其他 **NP** 轮有效协议)在严格意义上是否能被证明是零知识的(即,使用严格概率多项式时间模拟器)。

4.9.1 使用完全保密的承诺机制

为了清楚起见,我们先来详细地描述前面简要介绍的(图的三着色,$G3C$)常数轮交互证明。这个交互证明使用了两个不同的承诺方案。第一个是在构造 4.4.2 中提出的简单(完备绑定)承诺方案。我们用 $C_s(\sigma)$ 表示发送者使用硬币 s 对(三元)值 $\sigma\in\{1,2,3\}$ 的承诺。第二

个承诺方案是完备隐藏承诺方案(参见 4.8.2 小节)。为了简单起见,我们假定这个方案有一个承诺阶段,在这个承诺阶段中接收者发送一条消息给发送者,然后发送者回答一条消息(例如,构造 4.8.5)。让我们用 $P_{m,s}(\alpha)$ 表示当接收到(来自接收者的)消息 m 和使用硬币 s 时,发送者对字符串 α 的(完备隐藏)承诺。

构造 4.9.1($G3C$ 轮有效零知识证明):

- 公共输入:一个(可三着色的)简单图 $G=(V,E)$。令 $n \stackrel{\text{def}}{=} |V|$,$t \stackrel{\text{def}}{=} n \cdot |E|$,且 $V=\{1,\ldots,n\}$。
- 证明者的辅助输入:G 的一个三着色,记为 ψ。
- 证明者的预备步骤(P0):证明者调用完备隐藏承诺方案的承诺阶段,这将导致发送给验证者一条消息 m。
- 验证者的准备步骤(V0):验证者均匀独立地选择 t 条边的顺序,$\overline{E} \stackrel{\text{def}}{=} ((u_1,v_1),\ldots,(u_t,v_t)) \in E^t$,并发送给证明者一个对这些边的随机承诺。即,验证者均匀选择 $\overline{s} \in \{0,1\}^{\text{poly}(n)}$,并且将 $P_{m,\overline{s}}(\overline{E})$ 发送给证明者。
- 动机说明:现在,(从计算意义来说)验证者已经承诺为一个 t 条边的序列。因为这个承诺是完全保密的,所以证明者得不到关于这个边序列的任何信息。
- 证明者的第一步(P1):证明者均匀独立地选择 $\{1,2,3\}$ 上的 t 个置换 $\pi_1,\ldots,\pi_t$,并且对每个 $v \in V$ 及 $1 \le j \le t$,令 $\phi_j(v) \stackrel{\text{def}}{=} \pi_j(\psi(v))$。证明者使用这个(完备绑定,计算隐藏的)承诺方案来将自己承诺为对应于每个三着色的每个顶点的颜色。即,证明者平均独立地选择 $s_{1,1},\cdots,s_{n,t} \in \{0,1\}^n$,对每个 $i \in V$ 和 $1 \le j \le t$,计算 $c_{i,j}=C_{s_{i,j}}(\phi_j(i))$,并且将 $c_{1,1},\ldots,c_{n,t}$ 发送给验证者。
- 验证者的第一步(V1):验证者执行它的承诺的(标准)公布阶段,并产生序列 $\overline{E}=((u_1,v_1),\ldots,(u_t,v_t))$。即,验证者将 $(\overline{s},\overline{E})$ 发送给证明者。
- 动机说明:这时,验证者的所有承诺都公布了。现在验证者期望对于每个 j,第 j 个着色分配给顶点 u_j 和 v_j(序列 $\overline{E}$ 中第 j 个边的端点)的颜色。
- 证明者的第二步(P2):证明者检验从验证者那里刚刚接到的消息确实是在步骤 V0 中验证者所作承诺的有效公布,否则证明者立即中止。让我们用 $(u_1,v_1),\ldots,(u_t,v_t)$ 表示刚刚公布的 t 条边的序列。证明者使用完备绑定承诺方案的(标准)公布阶段来向验证者公布,对于每个 j,顶点 u_j 和 v_j 的第 j 个着色。即,证明者向验证者发送四元组的序列

$$(s_{u_1,1},\phi_1(u_1),s_{v_1,1},\phi_1(v_1)),\ldots,(s_{u_t,t},\phi_t(u_t),s_{v_t,t},\phi_t(v_t))$$

- 验证者的第二步(V2):验证者检查对于每个 j,在第 j 个四元组中的值是否构成承诺 $c_{u_j,j}$ 和 $c_{v_j,j}$ 的一个正确公布,并检查相应的值是否是不同的。即,在接收到 $(s_1,\sigma_1,s'_1,\tau_1)$ 到 $(s_t,\sigma_t,s'_t,\tau_t)$ 后,验证者检查对于每个 j,以下式子是否成立:$c_{u_j,j}=C_{s_j}(\sigma_j)$,$c_{v_j,j}=C_{s'_j}(\tau_j)$,且 $\sigma_j \neq \tau_j$(σ_j 和 τ_j 均在 $\{1,2,3\}$ 中取值)。如果所有条件都成立,那么验证者接受,否则验证者拒绝。

我们首先断言构造 4.9.1 确实是一个 $G3C$ 交互证明。显然,验证者总是接受 $G3C$ 中的公共输入。假定公共输入图 $G=(V,E)$ 不在 $G3C$ 中。使用证明者承诺的完备绑定特点,我们能够得到在步骤 P1 中承诺的值,并且可以说证明者在步骤 P1 中发送的每个"承诺颜色"包含至

少一条非法着色的边。使用步骤 V0 中验证者发送的承诺的完全保密性，我们得出，在步骤 P1 中证明者"无法知道"验证者想要看哪些边（即，就证明者拥有的信息而言，所有可能性都是等可能的）。因此，尽管证明者在接收到"边承诺"之后发送"着色承诺"，但是所有"承诺边"都具有合法的"承诺着色"的概率至多为

$$\left(1-\frac{1}{|E|}\right)^{t} \approx e^{-n} < 2^{-n}$$

（基本）模拟策略　现在我们继续说明构造 4.9.1 确实是零知识的（在允许期望多项式时间模拟器的不严格意义上）。对每个概率多项式时间交互机 V^*，我们引入一个将 V^* 作为黑盒子使用的期望多项式时间模拟器，记为 M^*。这个模拟器首先选择和固定一个 V^* 的随机磁带 r，并通过模拟证明者的预备步骤 P0 生成一个消息 m。给定输入图 G、随机磁带 r 和预备（证明者的）消息 m 的情况下，验证者 V^* 的承诺消息是确定的。因此，M^* 在输入 G、随机磁带 r 和消息 m 时，调用 V^*，并且得到相应的承诺消息，记为 CM。模拟器分两步运行。

S1. 提取质疑边：M^* 产生一个对哑值（dummy value）（例如，所有值等于 1）的 $n \cdot t$ 随机承诺的序列，并将其传送给 V^*。当 V^* 以正确公布 t 条边的序列，记为 $(u_1, v_1), \ldots, (u_t, v_t)$，进行应答时，模拟器记录下这些边并进入下一步。当 V^* 的应答并不是承诺消息 CM 的有效公布时，模拟器中止输出 V^* 的当前观点（例如，G, r, m 和对哑值的承诺）。

S2. 生成一个满足质疑边的交互（一个过于简单的说明）：令 $(u_1, v_1), \ldots, (u_t, v_t)$ 表示步骤 S1 中记录的边序列。机器 M^* 产生 $n \cdot t$ 个随机承诺 $(c_{1,1}, \ldots, c_{n,t})$ 的序列，使得对于每个 $j=1, \ldots, t$，$c_{u_j,j}$ 和 $c_{v_j,j}$ 是对 $\{1,2,3\}$ 中两个不同随机值的随机承诺，并且所有其他的 $c_{i,j}$ 是对哑值（例如，所有值等于 1）的随机承诺。这些潜在的值称为伪着色（pseudo-coloring）。模拟器将这个承诺序列传送给 V^*。如果 V^* 通过正确公布（先前记录的）边序列来应答，那么 M^* 能够实现对与 V^* 的"真实"交互的模拟（通过公布这些记录下来的边的顶点的颜色）。否则完全重复步骤 P2（直到成功）。

为了简单起见，我们在剩下的分析中忽略预备消息 m。而且，在剩下的分析中，我们也忽略在步骤 S1 和 S2 中调用时验证者公布两个不同的边承诺的可能性。不严格地说，这样的做法可以由下面的事实保证，即在期望多项式时间计算期间内，这种情况仅仅以可忽略不计的概率发生（因为如果不这样，就与验证者使用的承诺方案的计算明确性相冲突）。

过于简化的模拟器的运行时间　为了描述模拟器的行为，假定程序 V^* 总能正确公布步骤 V0 中所作的承诺。在这样的情况下，模拟器就会找出步骤 S1 中的质疑边，并且通过在步骤 S2 中利用它们，模拟器能够模拟 V^* 和真正验证者之间的交互。使用 4.4 节中的方法，能够证明模拟器同真实交互是计算不可分辨的。注意到在这样的情况下，模拟器的步骤 S2 仅执行一次。

现在来考虑一个更为复杂的情况：对于每个可能的内部硬币投掷序列 r，程序 V^* 正确公布在步骤 V0 中所作承诺的概率仅为 $\frac{1}{3}$。这一概率是在所有可能的（在模拟器步骤 S1 中）对哑值的承诺上取的。我们首先注意到，在接收到对（模拟器在步骤 V2 中产生的）伪着色序列的随机承诺之后，V^* 正确公布步骤 V0 中所作承诺的概率近似为 $\frac{1}{3}$（否则我们会得出与证明者所用承诺方案的计算安全性相矛盾）的结论。因此，模拟器进入步骤 S2 的概率为 $\frac{1}{3}$，并且步骤 S2 的每个执行成功完成的概率为 $p \approx \frac{1}{3}$。因此，步骤 S2 被执行的期望次数为 $\frac{1}{3} \cdot \frac{1}{p} \approx 1$。

现在让我们来考虑一般情况。令 $q(G,r)$ 表示对于输入图 G 和随机磁带 r，在接收到对（步骤

S1 中产生的)哑值的随机承诺之后,程序 V^* 正确公布在步骤 V0 中所作承诺的概率。同样地,我们用 $p(G,r)$ 表示(对于输入图 G 和随机磁带 r)在接收到对(模拟器在步骤 S2 中产生的)随机着色序列的随机承诺之后,程序 V^* 正确公布在步骤 V0 中所作承诺的概率。同前面一样,(就输入图 G 的大小而言)$q(G,r)$ 和 $p(G,r)$ 的区别是可以忽略的;否则会得到与证明者承诺方案的计算安全性相矛盾的结论。我们得到模拟器进入步骤 S2 的概率为 $q \stackrel{\text{def}}{=} q(G,r)$,步骤 S2 的每个执行成功完成的概率为 $p \stackrel{\text{def}}{=} p(G,r)$。因此,步骤 S2 被执行的期望次数为 $q \cdot \frac{1}{p}$。现在,有一条坏消息:我们无法确定 $\frac{q}{p}$ 是否近似等于 1,甚至无法确定 $\frac{q}{p}$ 是否由以输入大小为变量的多项式来限定(例如,令 $p=2^{-n}$,$q=2^{-n/2}$,那么它们之间的差别是可以忽略的,但是 $\frac{q}{p}$ 并不由 poly(n)限定)。这就是为什么模拟器的上述描述过于简单,并且确实需要进行修改的原因。

修改后的模拟器 我们通过对步骤 S2 做如下调整来使模拟器是期望多项式时间的。我们增加一个中间步骤 S1.5,仅当模拟器在步骤 S1 中未中止运行的时候执行。步骤 S1.5 的目的是提供对 $q(G,r)$ 的一个好的估计。这个估计通过重复步骤 S1 直到 V^* 正确公布的次数达到一个固定的数值($|G|$ 的多项式)计算出来(即,这个估计是成功次数与尝试次数的比率)。通过确定一个足够大的多项式,我们能够以极高的概率(即 $1-2^{-\text{poly}(|G|)}$)保证估计不超过 $q(G,r)$ 的常因子。容易验证,估计可在期望时间 $\text{poly}(|G|)/q(G,r)$ 内计算出来。对模拟器的步骤 S2 进行如下修改,对步骤 S2 的执行次数增加一个限定,并且如果这些执行没有一个能够产生 V^* 的正确公布,那么模拟器输出一个特殊的空交互。特别地,步骤 S2 被执行的次数最大为 $\text{poly}(|G|)/\tilde{q}$,其中 $\tilde{q}$ 是在步骤 S1.5 中计算出的 $q(G,r)$ 的估计。因此,修改后的模拟器有由 $q(G,r) \cdot \frac{\text{poly}(|G|)}{q(G,r)} = \text{poly}(|G|)$ 限定的期望多项式运行时间。

剩下的工作就是要分析修改后的模拟器的输出分布。我们通过限定修改后的模拟器输出一个特殊空交互的概率,来将分析约简为对原始模拟器输出的分析。这个概率等于

$$\Delta(G,r) \stackrel{\text{def}}{=} q(G,r) \cdot (1-p(G,r))^{\text{poly}(|G|)/q(G,r)}$$

我们认为 $\Delta(G,r)$ 是 $|G|$ 的可忽略函数。相反地,假定存在一个多项式 $P(\cdot)$、一个无限图序列 $\{G_n\}$ 和一个无限随机磁带序列 $\{r_n\}$,满足 $\Delta(G_n,r_n)>1/P(n)$。那么,对于每个这样的 n,我们都有 $q(G_n,r_n)>1/P(n)$。我们考虑如下两种情况。

情况 1:对于无穷多个 n,都有 $p(G_n,r_n) \geq q(G_n,r_n)/2$ 成立。在这种情况下,对于这些 n,我们得到

$$\begin{aligned}\Delta(G_n,r_n) &\leq (1-p(G_n,r_n))^{\text{poly}(|G_n|)/q(G_n,r_n)} \\ &\leq \left(1-\frac{q(G_n,r_n)}{2}\right)^{\text{poly}(|G_n|)/q(G_n,r_n)} \\ &< 2^{-\text{poly}(|G_n|)/2}\end{aligned}$$

这与我们的假定 $\Delta(G_n,r_n)>1/\text{poly}(n)$ 矛盾。

情况 2:对于无穷多个 n,都有 $p(G_n,r_n)<q(G_n,r_n)/2$ 成立。对于这些 n,我们有 $|q(G_n,r_n)-p(G_n,r_n)|>\frac{q(G_n,r_n)}{2}>\frac{1}{2P(n)}$,这与(证明者使用的)承诺方案的计算保密性矛盾。

于是,这两种情况都会产生矛盾。

结论 我们指出,可以修改构造 4.9.1,使得能够使用完备承诺方案的较弱形式。我们特别要提到的是具有完备后验保密性(参见 4.8.2 小节)的承诺方案。在这样的方案中,保密性仅仅在接收者公开它在承诺阶段中的硬币投掷时后验建立起来。在我们的情形中,证明者扮

演接收者的角色,验证者扮演发送者的角色。这对于建立后验保密性已经足够了,因为如果保密性没有建立起来,那么验证者将拒绝。在这种情况下不会有任何损害,因为完备承诺方案的保密性仅用来建立交互证明的合理性。因此,利用命题 4.8.8,我们得到如下推论。

推论 4.9.2:如果非均匀无爪集合存在,那么 $\mathcal{NP}$ 中的每种语言都有轮有效零知识证明系统。

4.9.2 限定欺骗证明者的能力

在无爪集合存在的假设下,构造 4.9.1 生成了 $\mathcal{NP}$ 轮有效零知识证明系统。使用看上去更一般的单向函数存在的假设,我们可以修改构造 4.9.1 以得到零知识计算合理证明系统。在修改的协议中,我们令验证者使用具有计算保密而不是构造 4.9.1 中的完备保密的承诺方案(因此,两个使用者都使用提供计算保密性的完备绑定承诺方案来承诺它们的消息)。而且,证明者使用的承诺方案还必须具有附加属性:在不"知道"承诺为什么值的情况下,进行承诺是不可行的。这样的承诺方案被称为是非遗忘的。我们通过定义和构造非遗忘的承诺方案来开始本节。

4.9.2.1 不可遗忘的承诺机制

承诺方案的非遗忘性与知识证明的定义(参见 4.7 节)紧密相关。

定义 4.9.3(非遗忘承诺方案):令(S,R)为定义 4.4.1 中所定义的(完备绑定)承诺方案。我们说承诺方案是**非遗忘的**,如果指定的接收者 R 构成一个对于如下关系总是信任 S 的知识验证者

$$\{((1^n,r,\overline{m}),(\sigma,s)):\overline{m}=\mathrm{view}_{R(1^n,r)}^{S(\sigma,1^n,s)}\}$$

其中,与定义 4.4.1 中一样,$\mathrm{view}_{R(1^n,r)}^{S(\sigma,1^n,s)}$表示对于输入 1^n 和本地硬币 r,在与交互机 S(具有输入$(\sigma,1^n)$,使用硬币 s)交互时,交互机 R 接收到的消息。

因此,接收者的指定程序 R 在承诺阶段结束时可能会接受或拒绝,并且假设这一决定能够反映发送者稍后产生承诺的合法公布(即成功完成公布阶段)的能力。我们强调,非遗忘性主要与欺骗性的发送者有关,因为指定的发送者稍后成功完成公布阶段并无困难(实际上,在承诺阶段,S 总能使接收者信服它的能力)。因此,任何能够使接收者接受的发送者程序(不仅是指定的 S)都能稍作修改,使其在承诺阶段结束时能够本地输出使其成功完成公布阶段的信息(即 σ 和 s)。修改后的发送者在期望时间内运行,这个期望时间与承诺阶段成功完成的概率成反比。

我们指出,在一般的承诺方案中,在承诺阶段结束时,接收者并不是一定要"知道"发送者稍后是否能成功执行公布阶段。例如,构造 4.4.2 中欺骗性的发送者能够(不被检测到地)执行承诺阶段,而不能成功地执行随后的公布阶段(即发送者可以只发送一个均匀选择的字符串)。这只能确保当发送者按照指定程序运行的时候,发送者总是能成功完成公布阶段。而且,对于构造 4.4.4 中提出的方案,欺骗性的发送者可以(不被检测到地)执行承诺方案,产生一个没有任何相应的合法公布的接收者观点(因此,公布阶段注定要失败);参考习题 14。尽管如此,我们还是能够证明如下定理。

定理 4.9.4:如果单向函数存在,那么具有常数轮通信轮数的非遗忘承诺方案也存在。而且,当并行执行多项式次时,承诺方案仍然保持保密特性。

多个副本的同时保密性对于 4.9.2.2 节中的应用至关重要。

证明思路：回想一下，(一般的完备绑定)承诺方案在单向函数存在的假设下是能够构造的(参见命题 4.4.5 和定理 3.5.12)。通过将这样的承诺方案与允许正常解除承诺的信息知识的零知识证明结合起来，我们可以得到非遗忘承诺方案(我们指出，在同样的假设下，这样的证明确实存在；参见 4.7 节)。然而，所生成的承诺方案的轮数是不定的(由于零知识证明的轮复杂度)，但我们需要的是轮数固定的承诺方案。我们好像进入了一个恶性循环，但有一种跳出的方法：我们可以使用知识的常数轮强证据不可分证明，而不使用(知识的)零知识证明。在同样的假设下，这样的证明确实存在：参见 4.6 节和习题 28。所生成的承诺方案具有额外属性：当并行执行多项式次时，保密特性在所有副本中同时保持。这个事实可以从(强)证据不可分辨证明的并行合成引理(参见 4.6 节)得到。■

4.9.2.2 修改构造 4.9.1

回想我们提到的对构造 4.9.1 的修改：验证者使用(具有计算保密性的)完备绑定承诺方案，而不是构造 4.9.1 中使用的具有完备保密性的承诺方案。另外，证明者所用的承诺方案是非遗忘的。

为适应我们当前的需要，我们采用第一种方法(即 4.9.1 小节中的方法)。我们首先指出，修改后的协议是 $G3C$ 计算合理性证明。和往常一样，验证修改后的协议满足完备性条件是很容易的。我们指出，修改后的协议并不满足(通常的)合理性条件(例如，一个具有指数级运算能力的“证明者”可以打破验证者的承诺，并产生随后会欺骗接收者接受的伪着色)。不过，可以证明修改后的协议确实满足(定义 4.8.1 中的)计算合理性条件。即，我们证明，对于每个多项式 $p(\cdot)$、每个多项式时间交互机 B、所有足够大的图 $G\notin G3C$，以及每个 y 与 z，都有：

$$\Pr[\langle B(y),V_{G3C}(z)\rangle(x)=1]\leq\frac{1}{p(|x|)}$$

其中 V_{G3C} 是修改后的协议中的验证者程序。

利用证明者使用的承诺方案的信息理论明确性，我们能够谈及由证明者的承诺推导出的惟一颜色分配。利用该承诺方案是非遗忘的这一事实，可以对证明者进行修改，使得在步骤 P1 中，证明者也(在它的本地输出磁带上)输出它在这个阶段承诺的值。由这个输出和验证者承诺方案的计算保密性，可以得到证明者产生的颜色分配与验证者的承诺几乎是独立的。因此，证明者能够欺骗验证者接受语言之外输入的概率至多略大于当证明者作出它的(颜色)承诺之后，验证者要求随机质疑时，相同事件的概率。这样就得到了(修改后)协议的计算合理性。(我们指出，在证明者使用并不确保是非遗忘的承诺方案的情况下，我们并不知道这一协议是否还是计算合理的[①]。)

① 特别地，我们不知道怎样排除这种可能性：在看到验证者在步骤 V0 的承诺之后，欺骗性的证明者能够在步骤 P1 中发送一些字符串，使得验证者公布它的承诺之后，证明者能够以一种合适的方式公开这些字符串。为了说明这个问题，假定双方希望用(完备绑定)承诺方案来投掷硬币，并且协议如下：首先，第一方承诺为一个比特，然后第二方承诺为一个比特，接着第一方公布它的比特，最后第二方公布它的比特，最终的结果是这两个公布的比特的 XOR。现在，通过复制第一方的消息，第二方能够强制结果总是为零！注意，当第二方使用非遗忘承诺方案时，这个问题不会出现。当第一方通过完备隐藏承诺方案进行承诺(并且第二方还是使用完备绑定承诺方案)时这个问题也不会出现。(后面的协议与 4.9.1 小节中提出的证明系统类似。)

说明(修改后)协议是零知识的甚至比第一种方法(即,4.9.1 小节中)的说明还要容易。原因是当说明这些协议的零知识特性时,我们使用了证明者承诺方案的保密性和验证者承诺方案的明确性。因此,只有承诺方案的这些特性才与协议的零知识特性相关。然而,当前(修改后的)协议使用的承诺方案具有的属性并不弱于构造 4.9.1 中所用的相应承诺方案的属性。特别地,就像构造 4.9.1 中证明者的承诺方案一样,修改后协议中的证明者承诺方案具有计算保密性。我们强调,像在构造 4.9.1 中证明者使用的简单承诺一样,这种承诺具有(多副本的)同时保密性。此外,修改后协议中的验证者承诺方案具有"信息理论的"明确性,而构造 4.9.1 中的验证者承诺方案仅具有计算明确性。因此,应用定理 4.9.4,我们有下面的推论。

推论 4.9.5:如果非均匀单向函数存在,那么 ***NP*** 中的每种语言都有一个轮有效零知识理论。

4.9.2.3　可选的构造

得到推论 4.9.5 的一种可选的方法是修改构造 4.4.7,使其允许容易的模拟,尤其是并行合成下的健壮性。这一修改的关键部分是具有"陷门特性"的承诺方案的概念。不严格地说,这样的承诺方案的承诺阶段包括接收者消息及随后的发送者消息,使得在给定接收者私有硬币的情况下,可有效产生与发送者的消息计算不可分辨的字符串,并且以后还能打开这些字符串来公布值。注意,这与计算绑定特性并不矛盾,因为后者与欺骗性的发送者有关(这些发送者并不知道接收者的私有硬币)。我们并不提出一个正式的定义,而只是简述一下这样的方案怎样构造和使用。

使用单向函数构造陷门承诺方案　令 f 是一个单向函数,令 $R_f \stackrel{\text{def}}{=} \{(f(w),w): w\in\{0,1\}^*\}$为一个 ***NP*** 关系(与 ***NP*** 集合 Range(f)对应)。在安全参数为 n 的情况下,接收者均匀选择 $r\in\{0,1\}^n$,利用标准约简将实例 $f(r)\in$ Range(f)简化为汉密尔顿圈(HC)问题的一个实例。所生成的图被发送给发送者,发送者(在不知道图中有一个汉密尔顿圈的情况下)被要求执行构造 4.7.14 中的步骤 P1,使得发送者能够对等于它的输入比特(发送者希望承诺为该比特)的步骤 V1 的消息作出响应。也就是说,为了承诺为比特 0,发送者发送一个该图的随机同构副本的邻接矩阵的承诺矩阵,而为了承诺为比特 1,发送者发送一个随机(简单)n 环路的邻接矩阵的承诺矩阵。因此,发送者的行为与构造 4.7.14 中的模拟器类似。这将被并行重复 n 次来生成一个计算隐藏(由于构造 4.7.14 中步骤 $P1$ 的证明者承诺)和计算绑定(因为否则发送者可以恢复 r 并对输入 $f(r)$求出 f 的逆)的固定轮承诺方案。相反地,r 的知识允许执行构造 4.7.14 中步骤 P1 的证明者策略,并且随后以两种方式中的任意一种来公开承诺。(注意,从 Range(f)到 HC 的标准简化通过一个映射来扩充,这个映射是从 f 的原像到相应的简化图中的汉密尔顿圈的映射,且该映射是多项式时间可计算的和可逆的。)

使用陷门承诺方案　使用前述方案的一种方法是将其用于构造 4.4.7 中的(步骤 P1 的)证明者承诺。出于这一目的,我们扩充陷门承诺方案,使得发送者在发送它的真实承诺(即,对应于构造 4.7.14 中步骤 P1 的消息)之前,我们让接收者证明它知道一个(相应的)陷门(即,产生它发给发送者的图的硬币序列)。这个知识证明仅需要是证据隐藏的,因此它能以固定轮数来执行。前面对构造 4.4.7 中修改的模拟器首先使用相应的知识提取器(来获得证明者承诺的陷门),然后利用这个陷门特性产生一些伪造的承诺,这些承诺以后能以需要的任何方式去公开(从而对验证者的要求进行应答)。

4.10* 非交互零知识证明

在本节中，我们考虑非交互零知识证明系统。这个模型包括 3 个实体：一个证明者，一个验证者和一个均匀选择的比特序列(可以被看作是由可信的第三方选择的)。验证者和证明者都可以读取这个随机序列，并且两者都可以额外投掷硬币。交互包括从证明者发送到验证者的单个消息，然后验证者做出决定(是否接受)。非交互零知识证明系统具有多种应用(例如，在对于选择明文攻击是安全的加密方案和签名方案中的应用)。

我们从允许我们证明预先限定长度的单个声明的基本定义和构造开始。接下来将处理的对象扩展为这样的证明系统：只要所有声明的总体长度是安全性参数的一个多项式(但是这个多项式又不是预先可知的)，就可以证明许多不同长度的声明。我们注意，与基本的处理不同，扩展处理允许我们证明总长度远远大于可信随机字符串长度的声明。可证明声明的总长度和可信随机字符串长度之间的关系，与被加密的消息(或签名文档)的总长度和加密密钥(或签名密钥)长度之间的关系类似。然而，我们强调，在当前的环境(非交互零知识证明)中，甚至处理基本的情况也是充满挑战的。

4.10.1 基本定义

非交互证明的模型从本质上看起来与 **NP** 证明模型相比，与通常的交互证明更接近。在某种意义上，**NP** 证明模型可以扩展，允许证明者和验证者使用一个公共的随机字符串和它们自己的硬币投掷。否则，与 **NP** 证明的情况一样，交互是最小的(即单向的：从证明者到验证者)。因此，在接下来的定义中，证明者和验证者都是普通的概率机器，除了公共输入外，它们也会得到一个均匀分布的(公共)参考字符串。我们强调，除了公共输入和公共参考字符串，证明者和验证者都可以投掷硬币并获得辅助输入。然而，为了简单起见，我们给出一个这些机器没有获得辅助输入(但它们仍然可以投掷附加硬币)的定义。验证者也将证明者产生的输出作为输入。

定义 4.10.1(非交互证明系统)：一对概率机器 (P,V) 叫做语言 L 的**非交互证明系统**，如果 V 是多项式时间的，并满足如下两个条件：

- 完备性：对于每个 $x\in L$，都有

$$\Pr[V(x,R,P(x,R))=1]\geq\frac{2}{3}$$

其中 R 是一个在 $\{0,1\}^{\text{poly}(|x|)}$ 上服从均匀分布的随机变量。

- 合理性：对于每个 $x\notin L$ 和每个算法 B，都有

$$\Pr[V(x,R,B(x,R))=1]\leq\frac{1}{3}$$

其中 R 是一个在 $\{0,1\}^{\text{poly}(|x|)}$ 上服从均匀分布的随机变量。

这个均匀选择的字符串 R 称为**公共参考字符串**。

和往常一样，两种情况下的错误概率通过足够多次地重复这个过程(使用多个独立选择的

参考字符串组成的序列），都可以（从$\frac{1}{3}$）减小到$2^{-\mathrm{poly}(|x|)}$。在声明合理性条件时，我们偏离了在R固定后允许$x\notin L$由对手选择的标准公式化描述；后面的合理性的"自适应"公式化描述在4.10.3.2节中使用，并且很容易将满足前面的（"非自适应"）合理性条件的系统变换为满足自适应合理性条件的系统（见4.10.3.2节）。

NP 中的每种语言都有非交互证明系统（在这些证明系统中没有使用随机性）。然而，这个 ***NP*** 证明系统不可能是零知识的（见定义4.10.2）。

非交互模型的零知识定义可以通过下面的事实简化：因为验证者不能影响证明者的行为，所以考虑单个的验证者（即，指定的验证者）观点的模拟就足够了。实际上，我们可以完全不考虑验证者（因为它的观点可以从公共参考字符串和证明者发送的消息产生出来）。

定义 4.10.2（非交互零知识）：语言L的非交互证明系统(P,V)是**零知识的**，如果存在一个多项式p和一个概率多项式时间算法M，使得总体$\{(x,U_{p(|x|)},P(x,U_{p(|x|)}))\}_{x\in L}$和总体$\{M(x)\}_{x\in L}$是计算不可分辨的，其中$U_m$是一个在$\{0,1\}^m$上服从均匀分布的随机变量。

这个定义也是"非自适应的"（即，公共输入不依赖于公共参考字符串）。在4.10.3.2节中给出并讨论了零知识的自适应公式化描述。

非交互零知识与常数轮零知识 我们强调，非交互零知识模型假定了验证者和证明者都可以使用的均匀选择的参考字符串的存在性。一个自然的建议是将这个假定替换为一个可以产生指定长度的均匀分布字符串的双方协议。这样的协议应该对双方中任意一方的对手行为是弹性的：即使双方中的一方偏离协议（使用任何概率多项式时间策略），输出都应该是均匀分布的。而且，看起来这样的协议应该具有一个强的模拟特性，允许为每个给定的输出产生一个随机执行副本。特别地，为了从非交互零知识证明中得到一个常数轮数零知识证明系统，似乎需要一个产生均匀分布字符串的常数轮数（强可模拟）协议。这样的协议可以用完备隐藏承诺方案来构造。与接下来的结果结合，可以推出另一个 ***NP*** 轮有效零知识证明的构造。

4.10.2 构造

一个假想的但是对非交互零知识证明系统很有帮助的抽象是隐藏比特模型。在这个模型中，公共参考字符串和以前一样是均匀选择的，但是只有证明者可以看到整个字符串。证明者发送给验证者的"证明"由两部分组成：一个"证书"和在公共参考字符串中某些比特位置的标识。验证者可以只检查公共参考字符串中由证明者指定的位置上的比特。当然，验证者也检查公共输入和"证书。"

定义 4.10.3（隐藏比特模型中的证明系统）：一对概率机器(P,V)叫做语言L的**隐藏比特证明系统**，如果V是多项式时间的，并满足如下两个条件：

- 完备性：对于每个$x\in L$，都有

$$\Pr[V(x,R_I,I,\pi)=1]\geq\frac{2}{3}$$

其中$(I,\pi)\stackrel{\text{def}}{=}P(x,R)$，$R$是一个在$\{0,1\}^{\mathrm{poly}(|x|)}$上服从均匀分布的随机变量，$R_I$是$R$在位置$I\subseteq\{1,2,\ldots,\mathrm{poly}(|x|)\}$上的子字符串。即，$R_I=r_{i_1}\cdots r_{i_t}$，其中，$R=r_1\cdots r_t$并且$I=(i_1,\ldots,i_t)$。

- 合理性：对于每个$x\notin L$和每个算法B，都有

$$\Pr[V(x,R_I,I,\pi)=1]\le\frac{1}{3}$$

其中$(I,\pi)\stackrel{\text{def}}{=}B(x,R)$，$R$ 是一个在$\{0,1\}^{\text{poly}(|x|)}$上服从均匀分布的随机变量，$R_I$ 是 R 在位置 $I\subseteq\{1,2,...,\text{poly}(|x|)\}$上的子字符串。

在两种情况下，I 都称为**公布比特集**，π 都称为**证书**。零知识的定义除了我们需要模拟$(x,R_I,P(x,R))=(x,R_I,I,\pi)$而不是模拟$(x,R,P(x,R))$外，与前面的定义相同。

与前面的声明一样，我们并不建议隐藏比特模型作为实际模型。模型的重要性源于两个事实。第一，这是一个简化了(其中的)证明系统的设计的“简明”模型；第二，隐藏比特模型中的证明系统可以容易地转换为非交互证明系统(即，实际的模型)。转换(其中使用了具有核心 b 的单向置换 f)如下。

构造 4.10.4(从隐藏比特证明系统到非交互证明系统)：令(P,V)是一个 L 的隐藏比特证明系统，假设 $f:\{0,1\}^*\to\{0,1\}^*$ 和 $b:\{0,1\}^*\to\{0,1\}$ 都是多项式时间可计算的。此外，令 $m=\text{poly}(n)$表示对于长度为 n 的公共输入，公共参考字符串的长度，并假设 f 是 1-1 的并且是等长度的。下面是一个非交互系统(P',V')的说明。

- 公共输入：$x\in\{0,1\}^n$。
- 公共参考字符串：$s=(s_1,...,s_m)$，其中每个 s_i 都在$\{0,1\}^n$ 中。
- 证明者(记为 P')：

1. 对于 $i=1,2,...,m$，计算 $r_i=b(f^{-1}(s_i))$；
2. 调用 P，得到$(I,\pi)=P(x,r_1\cdots r_m)$；
3. 输出(I,π,p_I)，其中对于 $I=(i_1,...,i_t)$，$p_I\stackrel{\text{def}}{=}(f^{-1}(s_{i_1})\cdots(f^{-1}(s_{i_t}))$。

- 验证者(记为 V')：给定证明者的输出$(I,\pi,(p_1\cdots p_t))$，验证者

1. 检验对于每个 $i_j\in I$ 是否有 $s_{i_j}=f(p_j)$(如果找到一个不匹配的，V'就中止并拒绝)；
2. 对于 $i=1,...,t$，计算 $r_i=b(p_i)$，并且令 $r=r_1,...,r_t$；
3. 对于(x,r,I,π)调用 V，并且当且仅当 V 接受时接受。

命题 4.10.5：令(P,V)，L，f，b 和(P',V')与构造 4.10.4 中定义的相同。如果 $\Pr[b(U_n)=1]=\frac{1}{2}$，那么$(P',V')$是一个 L 的非交互证明系统。而且，如果 P 是零知识的并且 b 是 f 的核心，那么 P'也是零知识的。

我们指出，即使 P 是完备零知识的，P'也不是完备零知识的。而且，即使 P 可以在多项式时间内实现(见下面的注释 4.10.6)，P'也不能在多项式时间内实现(甚至有辅助输入的帮助也不能)。

略证 为了说明(P',V')是 L 的一个非交互证明系统，我们注意到，均匀选择的 $s_i\in\{0,1\}^n$ 可以推导出均匀分布的比特 $r_i\in\{0,1\}$。这可以从 $r_i=b(f^{-1}(s_i))$，f 是 1-1 的和 $b(f^{-1}(U_n))\equiv b(U_n)$是无偏的这些事实得到。(注意，当 b 是 f 的核心时，它就几乎是无偏的(即，$\Pr[b(U_n)=1]=\frac{1}{2}\pm\mu(n)$，其中 μ 是一个可忽略的函数)。因此，只要说 b 是 f 的核心就足够了。)

为了说明 P'是零知识的，注意到，我们可以将 P 的一个有效模拟器转换为 P'的一个有效模拟器。特别地，对于数值 σ 的每个已经公布的比特，我们均匀选择一个字符串 $r\in\{0,1\}^n$ 满足 $b(r)=\sigma$，并将 $f(r)$放在公共参考字符串相应的位置上。对于每个未公布的比特，我们均匀选择一个字符串 $s\in\{0,1\}^n$，并将它放在公共参考字符串相应的位置上。P'模拟器的输出包

括:和前面一样产生的公共参考字符串,P'模拟器为 P 模拟器公布的比特产生的所有 r,P 模拟器的输出。运用 b 是 f 的核心这一事实,可以得到 P'模拟器的输出与验证者的观点(当接收一个来自 P'的证明时)是计算不可分辨的。■

注释 4.10.6(P'的有效实现):与前面声明的一样,在给定对 P 的黑盒访问的情况下,P'一般不能有效实现。需要的是(P')对 f 求逆的能力。另一方面,为了使 P'是零知识的,f 必须是单向的。明显的解决方法是使用陷门置换族,并让证明者知道这个陷门。此外,这个置换族应该具有其中的成员可以有效识别的特性(即,给定一个函数的描述,可以有效地判定它是不是在这个置换族中)。换句话说,P'通过在$\{0,1\}^n$ 上选择一个它知道陷门的置换 f 开始,并且像构造 4.10.4 中那样执行,惟一一点例外是它要在"证明"之后附加上 f 的描述。验证者根据证明中指定的函数 f 像构造 4.10.4 中那样执行。另外,它要检查 f 是否确实在置换族中。完备性和零知识条件都可以完全与命题 4.10.5 的证明一样得到。对于合理性条件,我们需要考虑置换族中所有可能的成员(不失一般性,最多有 2^n 个这样的置换)。对于每个这样的置换,讨论同前,因此通过累计讨论(如 4.10.3.2 节中使用的),可以得到我们的结论[①]。这个构造可以扩展为任何陷门置换;细节忽略。

现在我们开始讨论隐藏比特模型中证明系统的构造。特别地,我们将构造一个哈密尔顿环路(HC)问题的证明系统,其中哈密尔顿环路问题是 $\mathcal{NP}$ 完全问题(因此可以得到 $\mathcal{NP}$ 中任何语言的证明系统)。我们考虑有向图(和有向哈密尔顿环路的存在性)。下面,我们给出一个基本的零知识系统,在这个系统中哈密尔顿图被接受的概率为 1,而非哈密尔顿图被拒绝的概率为 $\Omega(n^{-3/2})$(这个系统建立在构造 4.7.14 中得到的系统的基础上)。

构造 4.10.7(HC 的隐藏比特系统):

- 公共输入:一个有向图 $G=(V,E)$,$n\stackrel{\text{def}}{=}|V|$。
- 公共参考字符串:被看作一个 $n^3\times n^3$ 的布尔矩阵 M,其中每一个元素是 1 的概率为 n^{-5}。

这通过将公共参考字符串分为多个长度为 $5\log_2 n$ 的块,并且当且仅当相应的块全是 1 的时候设置矩阵中相应的元素为 1 来实现。

- 定义:**置换矩阵**是其中每一行(和每一列)都只含有一个 1 的矩阵。**哈密尔顿矩阵**是一个置换矩阵,这个置换矩阵中所有行和所有列的 1 对应一个简单有向环路(即,相应的有向图有一个简单哈密尔顿环路)。

一个 $n^3\times n^3$ 的矩阵 M 称为是有用的,如果 M 含有一个广义的 $n\times n$ 哈密尔顿子矩阵,并且 M 中其他n^6-n^2 个元素都是 0。也就是说,一个有用的 $n^3\times n^3$ 矩阵恰含有 n 个 1,并且这 n 个 1 可以形成简单 n 环路$\{(\phi_1(i),\phi_2((i \bmod n)+1)):i=1,\ldots,n\}$,其中 ϕ_1 和 ϕ_2 是从$\{1,\ldots,n\}$到$\{1,\ldots,n^3\}$的 1-1 映射。

- 证明者:如果存在,令 C 是 G 中的一个哈密尔顿环路。证明者检验矩阵 M,并根据如下两种情况来动作。

情况 1:M 是有用的。令 H 表示 M 的 $n\times n$ 哈密尔顿子矩阵,C_H 是 H 中相应的哈密尔顿环路。

---

① 事实上,我们也需要重复(P,V)系统 O(n)次,以使得首次将合理性错误减小到$\frac{1}{3}\cdot 2^{-n}$。

- 证明者公布 M 中不在 H 中的所有(n^6-n^2)个元素。
- 证明者找到一个从 V 到 H 行的 1-1 映射 π_1，和一个从 V 到 H 列的 1-1 映射 π_2，使得 C 的边映射到 H 中为 1 的元素。

（G 中有向顶点对，不管是不是边，都以自然的方式进行映射。也就是说，(u,v)被映射到元素$(\pi_1(u),\pi_2(v))$上。要求映射对(π_1,π_2)是 C 到 C_H 的一个“同构”[①]。实际上，我们应该在 n 个可能的同构中指定一个同构。）

- 证明者公布不构成 G 的边的顶点对所对应的$(n^2-|E|)$个元素。

（这个对应是通过前面的映射实现的。也就是说，元素$(\pi_1(u),\pi_2(v))$被公布当且仅当$(u,v)\in V\times V\backslash E$。）

- 证明者输出映射对(π_1,π_2)（作为证书）。

总共有$(n^6-|E|)$个元素被公布，这些元素都是 0，而且证书是(π_1,π_2)。

情况 2：M 是无用的。在这种情况下，证明者公布 M 所有的元素。

（在这种情况下不需要证书。）

- 验证者：给定公布的元素，可能还有一个证书，验证者根据下面两种情况来动作。

情况 1：证明者没有公布 M 中所有的元素。令(π_1,π_2)是证明者发送/输出的证书。验证者检验 M 中所有的在(π_1,π_2)下在 E 中没有原像的元素是否已经都被公布而且确实都是 0。也就是说，如果矩阵中除了$\{(\pi_1(u),\pi_2(v)):(u,v)\in E\}$之外的所有元素都被公布，并且所有公布的比特都是 0，验证者就接受。

情况 2：证明者公布了 M 的所有元素。在这种情况下，当且仅当 M 是无用的时候，验证者就接受。

下面的事实对构造 4.10.7 的分析很有用。

事实 4.10.8：$\Pr[M\text{ 是有用的}]=\Omega(n^{-3/2})$。

略证　M 中 1 元素数目的期望值等于$(n^3)^2\cdot n^{-5}=n$。而且，矩阵恰好包含 n 个 1 元素的概率为 $\Theta(1/\sqrt{n})$。考虑 M 的任何一行，注意到这一行至多以概率$\binom{n^3}{2}\cdot(n^{-5})^2<n^{-4}$含有多于一个 1 元素。因此，以至少 $1-2n^3\cdot n^{-4}=1-O(n^{-1})$的概率，$M$ 的每行和每列都至多含有一个 1 元素。将这两个事实结合起来，就可以得到以概率 $\Omega(1/\sqrt{n})$，矩阵 M 含有一个 $n\times n$ 置换子矩阵，并且 M 中除了子矩阵之外的其他所有元素都为 0。现在注意到有 $n!$ 个$(n\times n)$置换矩阵，并且其中有$(n-1)!$ 个是哈密尔顿矩阵。因此，在 M 含有一个 $n\times n$ 置换子矩阵（其余元素都为 0）的条件下，矩阵 M 是有用的概率为 $1/n$。■

命题 4.10.9：存在一个图的哈密尔顿问题的（完备）零知识隐藏比特证明系统。而且，证明者可以通过一个将哈密尔顿环路作为辅助输入的多项式时间机器来实现。

略证　我们以描述构造 4.10.7 中验证者接受概率中的明显间隔来开始（为了满足要求，这个间隔可以通过对原证明系统的多项式次重复来扩大）。首先，我们指出，如果 G 具有哈密尔顿环路，并且证明者遵守程序，那么无论哪个矩阵 M 作为公共参考字符串，验证者都会接受。在情况 1 中这个结论成立是因为，映射对将 G 的哈密尔顿环路映射到 H 的哈密尔顿环路，而 H 仅包含 M 中的 1 元素，G 中的所有非边元素都映射到 M 的 0 元素。（在情况 2 中，结

---

① 阻止我们自由使用术语“同构”的细小技术细节是 H 并不是一个图。

论显然。)我们指出,当给定 G 的一个哈密尔顿环路作为辅助输入时,证明者的行为可以在多项式时间里执行。特别地,证明者要做的所有事情是检验 M 是否是有用的,并寻找两个给定 n 顶点环路之间的同构。

下面假设 G 是非哈密尔顿图,根据事实 4.10.8,矩阵 M 是有用的概率至少为 $\Omega(n^{-3/2})$。固定任何有用矩阵 M,我们要说明,不管证明者做什么,验证者都会拒绝 G。显然,如果证明者按照情况 2 中所述的进行动作,那么验证者就会拒绝(因为 M 是有用的)。因此,我们集中讨论证明者输出一对匹配(π_1,π_2)的情况(如情况 1)。令 H 表示 M 的(惟一的)$n\times n$ 哈密尔顿子矩阵,考虑下面两种子情况。

1. $\pi_1(V)\times\pi_2(V)$不等于 H。因为证明者必须公布所有不在子矩阵 $\pi_1(V)\times\pi_2(V)$中的元素,所以证明者必须公布 H 的某行或某列。但是这样的行或列必须包含一个 1 元素,这样验证者就会拒绝。

2. 否则,$\pi_1(V)\times\pi_2(V)=H$。同样,G 的每个非边都必须映射到 H 的一个 0 元素(否则验证者就会拒绝)。因此 H 中每个 1 元素的原像一定是 G 的一条边,这暗示了 G 具有一个哈密尔顿环路(与我们的假设相反)。

我们得出结论,在 G 是非哈密尔顿图的情况下,G 被拒绝的概率为 $\Omega(n^{-3/2})$。

最后,我们要说明证明者是零知识的。这通过构造一个模拟器来实现,这个模拟器在输入图 G 时,随机选择一个与公共参考字符串具有相同分布(即,每个元素为 1 的概率是 n^{-5})的 $n^3\times n^3$矩阵 M。如果 M 是无用的,那么模拟器就输出$(G,M,\{1,...,n^3\}^2)$(即,所有被公布的比特,M 中的值,并且没有证书)。否则,忽略这个(有用的)M,模拟器均匀选择一个 1-1 映射对(π_1,π_2),使得对于 $i=1,2$,有 $\pi_i:V\to\{1,...,n^3\}$。模拟器输出$(G,0^{n^6-|E|},I,(\pi_1,\pi_2))$,其中 $I\stackrel{\text{def}}{=}\{1,...,n^3\}^2\setminus\{(\pi_1(u),\pi_2(v)):(u,v)\in E\}$。读者可以很容易地验证模拟器的输出的分布与验证者看到的分布是相同的。■

应用命题 4.10.9 和 4.10.5 以及注释 4.10.6,我们可以得到如下结论。

定理 4.10.10:假设存在单向置换[①],那么 ***NP*** 中每种语言都有零知识非交互证明系统。此外,假设存在陷门置换族,那么 ***NP*** 中每种语言都具有一个零知识非交互证明系统,并且在这个系统中,证明者可以通过一个将 ***NP*** 证据作为辅助输入的概率多项式时间机器来实现。

4.10.3 扩展

我们给出在本节开始时提到的两个扩展:首先我们考虑当应用多项式次时(具有相同的公共参考字符串)还保持零知识的证明系统,然后我们考虑在公共参考字符串固定的情况下,声明(即公共输入)由对手选择时保持安全性的证明系统。

4.10.3.1 证明多个不同长度的声明

4.10.1 小节中给出的定义在两个方面被限制。首先,他们考虑仅对一个与公共参考字符串相关的声明的证明,而且允许公共参考字符串的长度大于声明的长度(尽管是声明长度的多项式)。下面将要给出的一个更强的定义允许 poly(n)个声明的证明,其中每个声明的长度都

---

① 与本章采用的惯例一样,在这里和后面,我们指困难性要求对于非均匀(多项式规模)电路也成立的构造。

是 poly(n)，并且都使用同一个 n 比特长的公共参考字符串。

我们首先注意到，只要处理声明的数量不受限制而每个声明的长度都预先限定的情况就足够了。特别地，对于任何 $\epsilon>0$，只要考虑 poly(n)个声明，每个的长度都是 n^ϵ，需要被证明是与同一个 n 比特长公共参考字符串相关的情况就足够了。这样做的原因是，我们可以以"零知识的方式"将任何长度为 poly(n)的 **NP** 声明简化为 poly(n)个 **NP** 声明的序列，每个的长度都是 n^ϵ：例如，首先我们将原始(poly(n)比特长的)**NP** 声明简化为一个关于 poly(n)个顶点的图的三着色的声明。接下来，我们使用承诺长度为 $n^\epsilon/2$ 的承诺方案承诺每个顶点的颜色。最后，对于每条边，我们(调用证明系统来)证明相应的两个承诺对应于取自{1,2,3}的两个不同的值。注意，每个这样的声明都是 **NP** 类型的，并且都涉及到一对 $n^\epsilon/2$ 比特长的字符串。

现在我们开始讨论实际的定义。首先我们指出，就非交互证明系统的定义(定义 4.10.1)而言，不需要做任何改变。我们仍然需要有能够被有效声明说服的能力，也需要防止错误声明的"保护"能力。但是，一个细微的技术区别是，在定义 4.10.1 中，我们用 n 表示声明的长度，并考虑长度为 poly(n)的公共参考字符串，在此处我们用 n 表示长度为 n^ϵ 的声明使用的公共参考字符串的长度。我们称 ϵ 是证明系统的基础常量。相反，零知识的定义不得不进行扩展以处理(预先指定的)无界证明序列。(注意，U_n 表示一个在$\{0,1\}^n$上服从均匀分布的随机变量。)

定义 4.10.11(非交互零知识，无界版本)：一个语言 L 的具有基础常量 ϵ 的非交互证明系统(P,V)是**无界零知识的**，如果对于每个多项式 p，都存在一个概率多项式时间算法 M，使得下面的两个总体是计算不可分辨的：

1. $\{((x_1,\ldots,x_{p(n)}),U_n,(P(x_1,U_n),\ldots,P(x_{p(n)},U_n)))\}_{x_1,\ldots,x_{p(n)}\in L_{n^\epsilon}}$
2. $\{M(x_1,\ldots,x_{p(n)})\}_{x_1,\ldots,x_{p(n)}\in L_{n^\epsilon}}$

这里 $L_l\stackrel{\text{def}}{=}L\cap\{0,1\}^l$。

我们指出，前面提出的非交互证明系统(例如，构造 4.10.4)不是无界零知识的；见习题 34。

我们现在讨论无界零知识(非交互)证明系统的构造。采用的思想是，通过潜在证明一个与公共参考字符串一部分相关的假象声明来强化模拟。这个将被潜在证明(关于这部分)的声明具有下面的特性：

1. 这个声明对于同样长度的字符串的一个可忽略部分也成立。因此，加上这个潜在的能力并不会显著影响合理性条件。

2. 满足声明的字符串与具有相同长度均匀分布的字符串是计算不可分辨的。因此，对于模拟来说，使用这样的字符串而不用均匀选择的字符串(在真正的证明系统中使用)是可以接受的。

3. 声明的决定问题在 **NP** 中。这就会简化为一个 **NP** 完全问题。

可以立即想到的一个关于字符串的声明由伪随机发生器产生。这将产生如下构造，其中 G 表示伪随机发生器。

构造 4.10.12(无界零知识非交互证明系统)：令 $G:\{0,1\}^l\rightarrow\{0,1\}^{2l}$，令 L_1 是一个 **NP** 完全语言，令 L 是任意一个 **NP** 语言，考虑下面的 **NP** 语言：

$$L_2\stackrel{\text{def}}{=}\{(x,p):x\in L\vee\exists w'\in\{0,1\}^{|x|}\ \text{s.t.}\ G(w')=p\}$$

考虑一个从 L_2 到 L_1 的标准简化，并令 q 是一个多项式，使得 L_2 的 $3l$ 比特长的实例映射到 L_1 的 $q(l)$比特长的实例。令(P,V)是 L_1 的一个普通的非交互证明系统，并假设对于某个

多项式 q'，系统 (P,V) 对于长度为 $q(l)$ 的声明使用一个长度为 $q'(l)$ 的公共参考字符串。假设 P 将一个 L_1 中成员的 **NP** 证据作为辅助输入，并令 $n=q'(l)+2l$。下面是 $L\in$ **NP** 的非交互证明系统的规范。

- 公共输入：$x\in\{0,1\}^l$。
- 公共参考字符串：$r=(p,s)$，其中 $p\in\{0,1\}^{2l}$，$s\in\{0,1\}^{n-2l}$。
- 证明者：

1. 运用从 L_2 到 L_1 的一个标准简化，证明者将 $(x,p)\in\{0,1\}^{l+2l}$ 简化为 $y\in\{0,1\}^{q(l)}$。此外，当给定一个 $x\in L$ 的 **NP** 证据 u 时，证明者将 u 简化[①]为一个 $y\in L_1$ 的证据 w。

2. 证明者对于公共输入 y、辅助输入 w 和公共参考字符串 s 调用 P，并获得它输出/发送的输出 π。

- 验证者：

1. 使用与 L_2 到 L_1 相同的标准简化将 (x,p) 简化为 y。

2. 对于公共输入 y、公共参考字符串 s 和证明者的输出 π，调用 V，并将 V 的判决作为自己的判决。

注意，简化将 L_2 的 $(l+2l)$ 比特长的实例映射到 L_1 的长度为 $q(l)$ 的实例。回忆前面，根据假设，证明系统 (P,V) 利用长度为 $q'(l)=n-2l$（这个长度实际上就是 s 的长度）的参考字符串来处理长度为 $q(l)$ 的 L_1 实例。令 $\varepsilon>0$ 是一个满足 $n^\varepsilon\leq l$（即 $(2l+q'(l))^\varepsilon\leq l$）的常数．那么我们有如下命题。

命题 4.10.13：令 (P,V) 同前，令 G 是一个伪随机发生器。此外，假设 P 是零知识的，并且当给定一个 **NP** 证据作为辅助输入时，P 可以在概率多项式时间里实现。那么构造 4.10.12 构成 L 的一个具有基础常量 ε 的无界零知识非交互证明系统。此外，证明者可以用一个将 **NP** 证据作为辅助输入的概率多项式时间机器来实现。

略证 新证明者的完备性和有效性可以立刻从 (P,V) 的假设中得到。注意到 p 在 G 的值域内的概率至多为 2^{-l}（取决于 (P,V) 的合理性），因此可以得到合理性条件。为了证明零知识特性，我们如下构造一个模拟器。这个模拟器均匀地选择 $u'\in\{0,1\}^l$ 和 $s\in\{0,1\}^{n-2l}$，设置 $P=G(u')$，并对每个在 L 个实例构成的序列中的实例 $x\in\{0,1\}^l$ 做如下处理：模拟器模拟证明者的程序（在输入 x 上），除了它使用 u' 作为 $(x,p)\in L_2$ 的 **NP** 证据。也就是说，模拟器将 $(x,p)\in L_2$ 简化为 $y'\in L_1$，同时将 **NP** 证据 u' 简化为证据 w'（对于 y'）。接下来，模拟器对于公共输入 y'、辅助输入 w' 和公共参考字符串 s 调用 P。因此，当给定实例序列 $\overline{x}=(x_1,\ldots,x_t)$ 时，模拟器输出 $(\overline{x},(p,s),P_{w'}(y'_1,s),\ldots,P_{w'}(y'_t,s))$，其中 y_i 是对 (x_i,p) 应用简化后的结果。注意，模拟器的效率取决于 P 的有效实现（和 G 的有效性）。为了证明模拟器的输出与验证者的观点是计算不可分辨的，我们将下面的两个观察合并起来（它们也依赖于 P 的有效实现）。

1. 公共参考字符串的分布实际上在两种情况下（即，真实的执行和模拟器的输出）是大不一样的。然而，根据伪随机发生器 G，这个区别是计算不可分辨的。因此，验证者在真实执行中的观点与它在公共参考字符串和模拟中选择的完全一样（但是证明者按照构造 4.10.12 动作）的情况下的观点是计算不可分辨的。

2. P 的零知识特性暗示了 P 是证据不可分辨的（见 4.6 节中的定义）。因此，不能区别出

---

① 我们又一次使用了标准简化结合了一个充足的证据简化的事实（参考习题 16）。

P 使用 $x \in L$ 的证据（同构造 4.10.12）的情况和 P 将伪随机序列 P 的种子作为证据使用（同模拟器做的一样）的情况。当重复这个证明过程多项式次时，这一点仍然满足。

换句话说，零知识的声明通过使用一个混合理论得到了证明，其中（单个）中间混合分量相当于对由模拟器产生的伪随机参考字符串（而不是真正随机的参考字符串）执行了证明者策略。这两个观察确立了这个中间混合分量与两个端点混合分量（这两者就是我们希望关联的两个总体）都是计算不可分辨的。■

使用定理 4.10.10 和命题 4.10.13，我们可以得到如下定理。

定理 4.10.14：假设陷门置换集合存在[①]，那么 $\mathcal{NP}$ 中每种语言都有一个无界零知识非交互证明系统。而且，证明者可以通过一个将 $\mathcal{NP}$ 证据作为辅助输入的概率多项式时间机器来实现。

4.10.3.2 适应性零知识

与 4.10.1 小节中提到的一样，迄今为止所用的定义都是非适应性的。这是指关于合理性条件和零知识条件（同样适用于完备性条件，但是因为所有通常使用的方案都具有理想的完备性[②]，所以就不关心这个问题）。在适应性零知识中，公共输入是在公共参考字符串固定以后由对手选择的。适应合理性的公式是很直接的，我们将读者的注意力集中到适应性零知识的公式上来。

定义 4.10.15（非交互零知识证明，适应性版本）：令 (P,V) 是语言 L 的一个非交互证明系统（即，与定义 4.10.1 中的相同）。

- 适应合理性：我们说 (P,V) 是**适应合理的**，如果对于每个 n 和每对函数 $\Xi:\{0,1\}^{\text{poly}(n)} \to (\{0,1\}^n \setminus L)$ 和 $\Pi:\{0,1\}^{\text{poly}(n)} \to \{0,1\}^{\text{poly}(n)}$，都有

$$\Pr[V(\Xi(R),R,\Pi(R)) = 1] \leq \frac{1}{3}$$

其中 R 是一个在 $\{0,1\}^{\text{poly}(n)}$ 上服从均匀分布的随机变量。

- 适应性零知识：我们说 (P,V) 是**适应零知识的**，如果存在两个概率多项式时间算法 M_1 和 M_2，满足对于每个函数 $\Xi:\{0,1\}^{\text{poly}(n)} \to (\{0,1\}^n \cap L)$，总体 $\{(R_n,\Xi(R_n),P(\Xi(R_n),R_n))\}_{n\in\mathbb{N}}$ 和 $\{M^{\Xi}(1^n)\}_{n\in\mathbb{N}}$ 是计算不可分辨的，其中 R_n 是一个在 $\{0,1\}^{\text{poly}(n)}$ 上服从均匀分布的随机变量，$M^{\Xi}(1^n)$ 表示下面随机化过程的输出：

1. $(r,s) \leftarrow M_1(1^n)$
2. $x \leftarrow \Xi(r)$
3. $\pi \leftarrow M_2(x,s)$
4. 输出 (r,x,π)

（也就是说，M_1 产生一个包括指定公共参考字符串 r 和 M_2 将要使用的辅助信息 s 的对 (r,s)。在给定一个适应选择的输入 x 和辅助信息 s 的情况下，M_2 产生一个所谓的证据 π。我们强调 x 可能依赖于 r，但是不依赖于 s）。

与前面一样，（在适应合理性条件中的）错误概率可以（从 $\frac{1}{3}$）减小到 $2^{-\text{poly}(|x|)}$。而且，任何非交互证明系统（即，非适应合理性的证明系统）都可以仅通过减少错误概率和应用联合界

---

① 见 198 页的脚注①。

② 也就是说，对于每个 $x \in L$，$Pr[V(x,R,P(x,R))=1]=1$ 实际上是成立的。

限转换为适应合理性的系统；也就是说，对于每个 $\Xi:\{0,1\}^{\mathrm{poly}(n)}\to(\{0,1\}^n\setminus L)$ 和 $\Pi:\{0,1\}^{\mathrm{poly}(n)}\to\{0,1\}^{\mathrm{poly}(n)}$，我们有

$$\begin{aligned}\Pr[V(\Xi(R),R,\Pi(R))=1] &\le \sum_{x\in\{0,1\}^n\setminus L}\Pr[V(x,R,\Pi(R))=1]\\ &\le 2^n\cdot\max_{x\in\{0,1\}^n\setminus L}\{\Pr[V(x,R,\Pi(R))=1]\}\end{aligned}$$

与前述的平凡转换(从非适应合理性到适应合理性的转换)相反，我们并不知道是否有从非交互零知识证明到适应性零知识证明的转换。然而幸运的是，4.10.2 小节中的说明可以扩展到适应性环境。(关键的思想是这些证明系统中的参考字符串可以由公共输入产生[①]。)我们可以得到如下定理。

定理 4.10.16：假设存在单向置换[②]，那么 *NP* 中的每种语言都有适应零知识的非交互证明系统。此外，假设存在陷门置换族，这样的证明系统中的证明者策略可以通过一个将 *NP* 证据作为辅助输入的概率多项式时间机器完成。

“进一步”论述扩展到一个允许适应性选择多项式个声明的模型(即，一个合并本节中讨论的两个扩展的模型)。

4.11* 多证明者零知识证明

在本节中我们考虑交互证明系统概念的一种扩展。特别地，我们考虑一个验证者同多个证明者(比方说，两个)的交互。证明者可以共享一个预先选择的策略，但是假定在它们和验证者交互期间，证明者之间不能互相交互。从直观上说，证明者能在验证者查询之前，而不是在查询期间调整它们的策略。多证明者模型令人联想起通常的警察办案过程，在这个过程中，将嫌疑犯隔离开并分别审讯。我们讨论这种模型适用的一个真实的(数字)环境。

多证明者交互证明的概念在复杂性理论中起着基础作用。但这不是我们现在要讨论的内容。在本节中我们仅讨论多方交互证明中关于零知识的内容。最重要的是，多证明者模型使得我们能够在不依赖任何复杂性理论假设的情况下构造 *NP*(完备)零知识证明系统。

4.11.1 定义

为简单起见，我们考虑两个证明者的模型。我们指出，更多证明者的使用并不会提供任何实质的优点(特别地，在本节中这也不能引起我们的兴趣)。不严格地说，两个证明者的交互证明系统是一个三方协议，在这个协议中，两方是证明者，第三方是验证者。这种模式中允许的惟一交互在验证者和每个单独的证明者之间进行。特别地，其中的一个证明者并不“知道”验证者发给另一个证明者的消息内容。但是，证明者确实共享一个“超出验证者影响范围”(就像

---

① 特别地，这对于命题 4.10.9 证明中的模拟器来说是很明显的。我们强调，这个模拟器判决所有与公共输入独立的隐藏比特的值(即，它们或者形成一个随机无用矩阵，或者它们“有效地”全为零)。命题 4.10.5 证明中的模拟器很容易修改成为这样的隐藏比特模型工作的模拟器。

② 见 198 页的脚注①。

单个证明者情况下那样)的随机输入磁带。两个证明者的模型是下面要介绍的两合作者(two-partner)模型的一个特例。

4.11.1.1 两合作者模型

两合作者模型包含两个与第三方(叫做单独方)交互的合作者。两个合作者可以事先就它们的策略达成一致,特别地,它们可以就公共均匀选择的字符串达成一致。然而,一旦和单独方的交互开始,合作者之间就再也不能交换信息。这样一种交互的如下定义是对定义 4.2.1 和定义 4.2.2 的扩展。

定义 4.11.1(两合作者模型):**两合作者模型**包括 3 个交互机,其中两个叫做**合作方**,第三个叫做**单独方**,它们之间的联系和交互说明如下。

- 三方使用相同的输入磁带,输入磁带的内容叫做公共输入。
- 两个合作者使用相同的随机磁带,这个随机磁带叫做**合作方随机磁带**(单独方使用单独的随机磁带)。
- 单独方有两对通信磁带和两个交换磁带,而不是只有一对通信磁带和一个交换磁带(如定义 4.2.1 中那样)。
- 两个合作者具有相同的标识,单独方具有相反的标识(参见定义 4.2.1 和 4.2.2)。
- 单独方的第一个(或第二个)交换磁带与第一个(或第二个)合作方的交换磁带相同,单独方的第一个(或第二个)只读通信磁带与第一个(或第二个)合作方的只写通信磁带相同,反之亦然。
- 在公共输入 x 下,三方的**联合计算**是一个三元组序列。每个三元组都包含这 3 个交互机的本地配置。每个合作方一单独方对之间的动作就像一对交互机的联合计算定义中那样。

我们用$\langle P_1,P_2,S\rangle(x)$表示对于公共输入 x,在与合作方 P_1 和 P_2 交互之后单独方 S 的输出。

4.11.1.2 两证明者的交互证明

现在可以根据一个证明者的情况(参见定义 4.2.4 和 4.2.6)来类似地定义两个证明者的交互证明系统。

定义 4.11.2(两证明者的交互证明系统):两合作者模型中的交互机三元组(P_1,P_2,V)叫做**语言 L 的证明系统**,如果机器 V(叫做验证者)是概率多项式时间的,并且下面的两个条件成立:

- 完备性:对于每个 $x\in L$,都有

$$\Pr[\langle P_1,P_2,V\rangle(x)=1]\geq\frac{2}{3}$$

- 合理性:对于每个 $x\notin L$ 和每对合作者(B_1,B_2),都有

$$\Pr[\langle B_1,B_2,V\rangle(x)=1]\leq\frac{1}{3}$$

一般来说,通过顺序重复执行协议足够多次,这两个条件下的误差概率能够(从$\frac{1}{3}$)减小到 $2^{-\mathrm{poly}(|x|)}$。但是在现在的环境中,通过并行重复来使误差减小的方法(一般来说)是有问题

的；参见本章结束处关于进一步阅读的建议。

（多证明者系统的）零知识概念与一个证明者情况下的概念完全一致。实际上，通过要求模拟器永远不会失败（即永远不会输出特殊符号⊥），我们能够使完备零知识的定义更为严格[①]。也就是说：

定义 4.11.3：我们说语言 L 的（两证明者）证明系统 (P_1, P_2, V) 是**完备零知识的**，如果对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，使得对于每个 $x \in L$，随机变量 $(P_1, P_2, V^*)(x)$ 和 $M^*(x)$ 同分布。

扩展为辅助输入（零知识）模型是直接的。

4.11.2　两发送者的承诺方案

本节的目的是获得一个构造 $\mathcal{NP}$ 中每种语言的完备零知识两证明者证明系统的方法。这种方法的核心使用了保持信息理论保密性和明确性（即，完备隐藏性和完备绑定性）的两发送者一接收者的承诺方案。我们强调，在单发送者模型下，不可能同时实现信息理论保密性和明确性。

4.11.2.1　定义

不严格地说，两发送者承诺方案是两合作者模型的一个有效的两阶段（*two-phase*）协议，通过这个协议，叫做发送者的两合作者能够将它们承诺为一个值，使得能够满足如下两个互相冲突的要求：

1. 保密性：在承诺阶段结束时，单独方（叫做接收者）并没有获得关于发送者的值的任何信息。

2. 明确性：假定承诺阶段成功完成，那么如果随后发送者能够执行公布阶段，使得接收者以概率 p 接受数值 0，则它们不能执行公布阶段使接收者接受数值 1 的概率远大于 $1-p$。我们强调，在整个承诺和公布阶段中不允许两个发送者之间有任何交互。（我们指出，对于每个 p，发送者总能执行承诺阶段使得它们随后能够以概率 p 公布 0，以概率 $1-p$ 公布 1；参见习题 35。）

我们不是给出一个一般性的定义，而只是将我们的注意力集中在只有第一个发送者（和接收者）参与承诺阶段，只有第二个发送者参与（规范）公布阶段的两发送者承诺方案的特例。而且，不失一般性，我们假定在公布阶段，第二个发送者将（第一个发送者在承诺阶段中使用的）联合随机磁带的内容发送给验证者。我们再次强调，在整个承诺和公布的过程中，两个发送者之间不能交换信息；因此，在承诺阶段第二个发送者不知道接收者发送给第一个发送者的消息。

定义 4.11.4（两发送者的比特承诺）：**两发送者的比特承诺方案**是两合作者模型的一个满足如下条件的概率多项式时间交互机的三元组，记为 (S_1, S_2, R)。

- 输入规范：公共输入是一个一元的整数 n，叫做**安全参数**。两个合作者叫做**发送者**，有一个辅助私有输入 $v \in \{0,1\}$。
- 保密性：0 承诺和 1 承诺是同分布的。也就是说，对于每一个与第一个发送者（即 S_1）交

---

① 回想在定义 4.3.1 中，允许模拟器（以至多 $\frac{1}{2}$ 的概率）失败。

互的概率(并不一定是多项式时间的)机器 R^*，随机变量 $\langle S_1(1),R^*\rangle(1^n)$ 和 $\langle S_1(0),R^*\rangle(1^n)$ 是同分布的。

● 明确性：预备知识：为简单起见，$v\in\{0,1\}$ 和 $n\in\mathbb{N}$ 隐含在所有概念中。

1. 如定义 4.4.1 所述，接收者对与(第一个)发送者交互的观点记为 $(r,\overline{m})$，包括接收者使用的随机硬币(记为 r)和从(第一个)发送者接收的消息序列(记为 $\overline{m}$)。

2. 令 $\sigma\in\{0,1\}$。我们说字符串 s 是接收者观点 $(r,\overline{m})$ 的一个可能的 σ 公开(**possible σ -opening**)，如果 $\overline{m}$ 表示当 R 使用本地硬币 r 并且同使用本地硬币 s 和输入 $(\sigma,1^n)$ 的 S_1 交互时所接收的消息。

3. 令 S_1^* 为第一个发送者的任意程序。p 为一个实数，$\sigma\in\{0,1\}$。我们说 p 是一个接收者对与 S_1^* 交互的观点的一个 σ 公开的概率上界，如果对于每个与接收者的硬币投掷统计独立的随机变量 X(表示公布阶段中第二个发送者所发送的字符串)，X 是接收者对与 S_1^* 交互的观点的一个可能的 σ 公开的概率至多为 p。也就是说，

$$\Pr[X\text{ 是 }\langle S_1^*,R\rangle(1^n)\text{ 的一个 }\sigma\text{ 公开}]\leq P$$

(这个概率是在接收者的硬币投掷，策略 S_1^* 和随机变量 X 上取的)。

4. 令 S_1^* 如上所述，对于每个 $\sigma\in\{0,1\}$，令 P_σ 为接收者对与 S_1^* 交互的观点的一个 σ 公开的概率上界。如果 $p_0+p_1\leq 1+2^{-n}$，那么我们就说**接收者对与 S_1^* 交互的观点是明确的。**

明确性要求指出，对于第一个发送者的每个程序 S_1^*，接收者与 S_1^* 的交互是明确的。

在明确性要求的公式化描述中，随机变量 X 表示第二个发送者可能的策略。这个策略可能依赖于两个发送者共享的随机输入，但是独立于接收者的随机硬币(因为这些硬币的信息，如果有的话，只发送给第一个发送者)。对于接收者的每个可能的硬币投掷，两个发送者采用的策略决定了它们在 0 公开和 1 公开中的成功概率组成的概率对(实际上，这些概率的界限仅由第一个发送者的策略决定)。明确性条件指出，这样的概率对在所有可能的接收者硬币投掷上取的均值是一个加起来最多为 $1+2^{-n}$ 的概率对。从直观上看，这意味着，发送者们所构成的危害不能超过随机决定是承诺 0 还是承诺 1。保密性要求和明确性要求都是信息理论的(从这个意义上说，对对手的策略在计算上不做限制)。我们强调，我们已经隐含假定了公布阶段采取如下的标准形式。

1. 第二个发送者发送给接收者初始的私有输入 v 和第一个发送者在承诺阶段使用的随机硬币 s。

2. 接收者验证 v 和 s(以及 R 在承诺阶段中使用的私有硬币(即 r))确实产生了 R 在承诺阶段中接收到的消息。验证(通过运行程序 S_1 和 R)在多项式时间内完成。

考虑分配给定义 4.11.4 明确性条件中的每个策略 S_1^* 的对 (p_0,p_1)。我们注意到，p_0+p_1 最大的可能值通过两发送者的决定策略是可以达到的[①]。因此，考虑第一个发送者的任意决定策略 S_1^* 和对于 $\sigma\in\{0,1\}$ 的一个确定的 σ 公开，记之为 s^σ，就足够了。因此，明确性条件

---

① 我们使用平均的方法。首先注意到，对于每个(概率)S_1^* 和 σ，都存在一个字符串 s^σ，使任何固定的字符串是 $\langle S_1^*,R\rangle(1^n)$ 的一个 σ 公开的概率最大。因此，s^σ 是 $\langle S_1^*,R\rangle(1^n)$ 的公开的概率是 X(见定义中)是 $\langle S_1^*,R\rangle(1^n)$ 的 σ 公开的概率的上界。同样地，固定这样的对 (s^0,s^1)，我们将 S_1^* 看作是在第一个发送者的决定策略上的分布，并且考虑分配给每个这样的策略 S_1^{**} 的两个概率之和。因此，存在这样一个决定策略 S_1^{**}，对于这个决定策略，概率和至少与 S_1^* 的相应概率和一样大。

是说，对于每个这样的 S_1^*，s^0 和 s^1，

$$\sum_{\sigma\in\{0,1\}} \Pr[s^\sigma \text{ 是}\langle S_1^*, R\rangle(1^n)\text{ 的 }\sigma\text{ 公开}] \le 1+2^{-n}$$

事实上，对于下面将要提出的构造，我们将建立一个更强的条件。

强明确性条件：对于每个决定策略 S_1^* 和每个字符串对(s^0,s^1)，都有

$$\Pr[\forall\sigma\in\{0,1\}, s^\sigma \text{ 是}\langle S_1^*, R\rangle(1^n)\text{ 的 }\sigma\text{ 公开}] \le 2^{-n}$$

（显然，如果不满足明确性条件，那么也不满足强明确性条件。）

4.11.2.2 一个构造

根据前面的约定，只要明确描述承诺阶段（在承诺阶段中，只有第一个发送者参与）就足够了。

构造 4.11.5（一个两发送者的比特承诺）：

- 预备知识：令 π_0 和 π_1 表示两个在$\{0,1,2\}$上的固定置换，使得 π_0 为恒等置换，π_1 为包括一个单个位置转换，比如说(1,2)的置换。也就是说，$\pi_1(1)=2,\pi_1(2)=1,\pi_1(0)=0$。
- 共同输入：安全参数 n（以一元方式表示）。
- 发送者的输入：$\sigma\in\{0,1\}$。
- 约定：假定发送者随机磁带的内容编码为均匀选择的$\bar{s}=s_1\cdots s_n\in\{0,1,2\}^n$。
- 承诺阶段：

1. 接收者均匀选择$\bar{r}=r_1\cdots r_n\in\{0,1\}^n$，并将$\bar{r}$发送给第一个发送者。
2. 对于每个 i，第一个发送者计算 $c_i \stackrel{\text{def}}{=} \pi_{r_i}(s_i)+\sigma \bmod 3$，并将 $c_1\cdots c_n$ 发送给接收者。

我们指出，如果第二个发送者已经知道（接收者发送给第一个发送者的）$\bar{r}$，那么它就可以以任何一种方式打开承诺。关键是第二个发送者不知道$\bar{r}$，这个事实彻底限制了它的欺骗能力。

命题 4.11.6：构造 4.11.5 构成一个两发送者的比特承诺方案。

证明：通过观察对于每个$\bar{r}\in\{0,1\}^n$ 的选择，第一个发送者发送的消息在$\{0,1,2\}^n$ 上服从均匀分布，就可以得到保密性特性。

用反证法证明（强）明确性特性。作为一个动机，我们首先考虑当 n 等于 1 时前述协议的执行，然后说明对于两个发送者来说，以两种方式都总能公开承诺是不可能的。考虑任意对(s^0,s^1)，其中 s^0 是一个可能的 0 公开，s^1 是一个可能的 1 公开，这两者都是就接收者的观点而言的。我们强调，这些 s^σ 必须与所有可能的接收者观点相匹配（否则公开就并不总能成功）。因此，对于每个 $r\in\{0,1\}$，$\pi_r(s^0)$和 $\pi_r(s^1)+1(\text{mod}3)$都必须与接收者所接收的作为它发送的消息 r 的回应的消息相符合。因此，对于每个 $r\in\{0,1\}$，$\pi_r(s^0)\equiv\pi_r(s^1)+1(\text{mod}3)$都成立。这样就得到了矛盾，因为不存在 $s^0,s^1\in\{0,1,2\}$能够同时满足 $\pi_0(s^0)\equiv\pi_0(s^1)+1 \pmod 3$和 $\pi_1(s^0)\equiv\pi_1(s^1)+1 \pmod 3$，理由是第一个等式隐含有 $s^0\equiv s^1+1 \pmod 3$，与第二个等式联合起来，就可以得出 $\pi_1(s^1+1 \bmod 3)\equiv\pi_1(s^1)+1 \pmod 3$，然而对于每个 $s\in\{0,1,2\}$，都有 $\pi_1(s+1 \bmod 3)\not\equiv\pi_1(s)+1 \pmod 3$。

我们现在回到强明确性特性的实际证明中来。第一个发送者的任意（决定）策略可以用一个函数刻划，该函数记为 f，将 n 比特长的字符串映射为$\{0,1,2\}^n$ 中的序列。因此，当使用硬币序列$\bar{r}=r_1\cdots r_n\in\{0,1\}^n$ 时，接收者的观点包括$(\bar{r},f(\bar{r}))$。令$\bar{s}^0$ 和$\bar{s}^1$ 表示第二个发送者的任

意公开尝试(即,分别对应于 0 公开和 1 公开)。不失一般性,我们假定$\bar{s}^0$ 和$\bar{s}^1$ 都在$\{0,1,2\}^n$ 中,并且令$\bar{s}^\sigma=s_1^\sigma\cdots s_n^\sigma$(其中 $s_i^\sigma\in\{0,1,2\}$)。强明确性特性指出,对于一个均匀选择的$\bar{r}\in\{0,1\}^n$,$\bar{s}^0$ 和$\bar{s}^1$ 分别是接收者观点$(\bar{r},f(\bar{r}))$的 0 公开和 1 公开的概率至多为 2^{-n}。

让我们用 R^σ 表示序列$\bar{s}^\sigma$ 是接收者观点$(\bar{r},f(\bar{r}))$的一个可能 σ 公开的所有字符串$\bar{r}\in\{0,1\}^n$ 构成的集合。即,

$$R^\sigma=\{\bar{r}:(\forall i)\ f_i(\bar{r})\equiv\pi_{r_i}(s_i^\sigma)+\sigma\pmod 3\}$$

其中$\bar{r}=r_1\cdots r_n$,$f(\bar{r})=f_1(\bar{r})\cdots f_n(\bar{r})$。那么由强明确性特性可知 $|R^0\cap R^1|\leq 2^{-n}\cdot|\{0,1\}^n|$。即:

结论 4.11.6.1: $|R^0\cap R^1|\leq 1$。

证明: 相反,我们假定$\bar{\alpha},\bar{\beta}\in R^0\cap R^1$(且$\bar{\alpha}\neq\bar{\beta}$)。那么存在一个 i,使得 $\alpha_i\neq\beta_i$,并且不失一般性,$\alpha_i=0$($\beta_i=1$)。由 R^σ 的定义有

$$f_i(\bar{\alpha})\equiv\pi_0(s_i^0)\pmod 3$$
$$f_i(\bar{\alpha})\equiv\pi_0(s_i^1)+1\pmod 3$$
$$f_i(\bar{\beta})\equiv\pi_1(s_i^0)\pmod 3$$
$$f_i(\bar{\beta})\equiv\pi_1(s_i^1)+1\pmod 3$$

正如动机讨论中所谈到的那样,出现了矛盾。也就是说,利用前两个等式和 π_0 是一个恒等置换的事实,我们有 $s_i^1+1=s_i^0\pmod 3$,将这一结果同后两个等式联合起来,可得

$$\pi_1(s_i^1+1)=\pi_1(s_i^0)\equiv\pi_1(s_i^1)+1\pmod 3$$

这与(容易验证的)事实:对于每个 $s\in\{0,1,2\}$,都有 $\pi_1(s+1\mod 3)\neq\pi_1(s)+1\pmod 3$矛盾。

该命题证毕。■

注释 4.11.7(并行执行)。这个证明可扩展到协议的多个实例并行执行的情况。特别地,通过构造 4.11.5 的 t 个并行执行,我们可得一个关于 t 比特长字符串的两发送者的承诺方案。注意,我们对得到验证者的观点有两个相互冲突的公开的概率至多为 2^{-n}(或者甚至是 $t\cdot 2^{-n}$)感兴趣,而不是对寻求误差减小(即,以 $2^{-t\cdot n}$为界的概率)感兴趣。

4.11.3 NP 完备零知识

通过修改构造 4.4.7,可以很容易得到 NP 中任何语言的两证明者完备零知识证明系统。这一修改包括用构造 4.11.5 中的两发送者比特承诺方案来代替构造 4.4.7 中所用的比特承诺方案。特别地,修改后的图着色证明系统介绍如下。

图着色的两证明者原子证明

1. 第一个证明者使用证明者的随机磁带决定着色置换。为了承诺为结果中的每种颜色,第一个证明者调用两发送者比特承诺(的承诺阶段),同时将安全参数设为图的顶点数。(第一个证明者起着第一个发送者的作用,而验证者起接收者的作用。)

2. 验证者均匀选择一条边,并将其发送给第二个证明者。作为应答,第二个证明者通过发送承诺阶段相应实例使用的证明者随机磁带的相关部分,来公布要求的边的顶点的颜色。

通常,我们可以知道,证明者总能使验证者相信有效结论(即,完备性条件成立)。利用两

发送者承诺方案的明确性特性(并且忽略同“理想情况”的误差 2^{-n}),我们可以考虑第一个证明者以任意概率分布随机选取一个图的顶点的颜色分配。我们强调,即使承诺协议的多个实例同时执行(参见注释 4.11.7),这个结论仍然成立。如果这个图不是可三着色的,那么第一个证明者所选择的每种可能的颜色分配都是不合法的,这样就得到了弱合理性特性。然而,多项式次执行这个协议,即使是并行的,我们都能生成一个满足合理性要求的协议。我们强调,并行处理(作为减少错误概率的方式)在这里是有效的来源于两发送者承诺方案的明确性,而不是来源于通用的“并行合成引理”(这在两证明者的环境中是高度非平凡的)。

现在我们回到这个协议的零知识性上。事实证明,接下来要介绍的部分比我们以前所介绍的要容易处理得多。在模拟器的构造中,我们利用如下事实:模拟器起到两个证明者的作用(因此承诺方案的明确性不适用)。特别地,扮演两个证明者角色的模拟器能够容易地以它想要的任何方式公开每个承诺(这里我们利用了构造 4.11.5 中的承诺方案的特定结构)。细节如下。

图着色原子证明的模拟

1. 模拟器产生随机的“无承诺(commitments to nothing)”。也就是说,模拟器调用验证者,并且通过在{0,1,2}上均匀选择的字符串序列来应答属于承诺阶段的验证者消息。

2. 一旦从验证者那里接收到质疑边(u,v),模拟器就均匀选择两种不同的颜色 ϕ_u 和 ϕ_v,并且公开相应的承诺以公布这些值。模拟器这样做没有困难,因为不像第二个证明者,模拟器知道承诺阶段中验证者发送的消息。特别地,给定接收者关于承诺阶段的观点$(r_1\cdots r_n, c_1\cdots c_n)$,通过对所有的 i 设置 $s_i=\pi_{r_i}^{-1}(c_i)$(或 $s_i=\pi_{r_i}^{-1}(c_i-1)$),就可以计算 0 公开(或 1 公开)。注意,接收者关于承诺阶段的观点等于验证者和第一个证明者交换的消息,并且这些消息都在步骤 1 中产生。

注意,模拟器的消息与真实交互中证明者的消息同分布。(惟一的区别是这些消息的产生方式:在真实交互中,s_i 是在{1,2,3}中均匀选择的,并且(同以 r_i 及随机置换的着色一起)决定了 c_i;而在模拟中,c_i 是在{1,2,3}中均匀选择的,并且(同 r_i 以及{1,2,3}中的一个随机对一起)决定了所公布的 s_i。)

我们指出,所有的讨论都可以很容易地扩充到多项式个协议实例并行执行的情况。因此,我们得到下面的定理。

定理 4.11.8:***NP*** 中的每种语言都有一个完备零知识两证明者证明系统。此外,这个证明系统还有如下附加特性。

- 通信过程只在一轮中完成:验证者只发送一条消息给两个证明者的每一个,这两个证明者依次只发一条消息作为应答。
- 合理性错误指数级减小。
- 两个证明者的策略可以通过将 ***NP*** 证据作为辅助输入的概率多项式时间机器来实现。

有效性的改进 通过应用概率可检验证明(PCP)的结果,***NP*** 的两证明者(完备)零知识证明的有效性可获得惊人的改进。特别地,这样的证明系统具有可忽略的误差概率,且可在概率多项式时间内实现,使得交互中所交换的总比特数是多项式对数的。

4.11.4 应用

多证明者交互证明仅在“证明实体”可被“分割”成两个(或更多的)部分,并且每部分之间

在整个证明过程中保持互不知情的情况下有用。在这样的情况下，我们不用依靠复杂性理论的假设就可得到完备零知识证明。换句话说，通常的（公认的）困难性假定可被证明过程发生的特定环境的物理假定取代。

一个自然的应用是关于身份认证的问题，特别是在某个终端上的用户的身份认证问题。在 4.7 节中，我们讨论了怎样将身份认证简化为（某种 *NP* 关系的）知识的零知识证明。这里，我们建议给每个用户提供两张智能卡，来实现知识的两证明者零知识证明中的两个证明者。这两张智能卡必须插入同一终端的两个不同插槽，这就确保了两张智能卡不能互相通信。这个终端在知识的零知识证明中起到验证者的作用。因此，这个终端可有效防止冒充，而用户可以防止非法终端企图从智能卡中提取信息（从而只有用户自己的代理才能冒充他）。

4.12　其　　他

4.12.1　历史记录

交互证明系统是由 Goldwasser，Micali 和 Rackoff[124][①]提出的。交互证明的一个受限形式叫做 Arthur-Merlin 游戏（或公开硬币证明），在[8]中进行了介绍，并在[128]中被证明与通用交互证明等价。非同构图的交互证明是 Goldreich，Micali 和 Wigderson 提出的[112]。交互证明的令人吃惊的定理证明能力在[157，198]中进行了描述，并分别说明了 co*NP* 交互证明以及（更一般的）***PSPACE*** 交互证明。

零知识的概念由 Goldwasser，Micali 和 Rackoff 在同一篇论文[124]中提出。这篇论文还包含一个二次非剩余的完备零知识证明。图同构的完备零知识证明系统是 Goldreich，Micali 和 Wigderson 提出的[112]。

使用任何（非均匀安全）承诺方案的 *NP* 中所有语言的零知识证明系统也是由 Goldreich，Micali 和 Wigderson 提出的[112][②]。（*IP* 中所有语言的零知识证明系统已经在[136]和[25]中给出。）

零知识证明在密码学上的应用是在[124]中最先引入这一动机的。零知识证明可应用于解决[81]和[54]中的密码学问题。然而，一旦说明如何构造关于 *NP* 中每种语言的零知识证明系统，越来越多的应用就变得可能。特别地，在[112，113]中已经出现了密码学协议构造的一般方法。

基于单向置换的承诺方案的构造可追溯到[31]。基于伪随机发生器的承诺方案的构造是由 Naor 提出的[170]。

关于高级章节的评价

---

① 他们论文的最早版本可追溯到 1983 年初。但是这篇文章在被主要的学术会议拒绝过 3 次之后，首次发表是在 1985 年，同时发表的还有 Babai 的论文[8]。

② 随后一个更弱一些的结果出现在[41]中：它利用更强的困难问题假设（特别地，二次剩余问题的困难性），给出了一个 *NP* 零知识证明系统的可选构造。

负面结果 描述随机性的必要性和零知识证明的交互的结果来自[115]。提供具有几乎完备零知识证明(即,定理 4.5.8)语言复杂性上界的结果来自[83]和[2]。显示单向函数对非平凡零知识是必要的的结果来自[181]。关于零知识证明系统(即,命题 4.5.9 和定理 4.5.11)并行合成的负面结果来自[106]。

证据不可分辨性 证据不可分辨性和证据隐藏的概念是由 Feige 和 Shamir[78]提出和发展的。4.6 节就是以他们的工作为基础的。

知识证明 知识证明的概念源自 Goldwasser,Micali 和 Rackoff 的论文[124]。想要给出这个概念的定义的更早尝试出现在[75]和[205]中;然而,那些定义都不能令人满意。与知识证明的定义相关的问题由 Bellare 和 Goldreich 进行了深入研究[17],我们遵循他们的定义。知识的零知识证明在身份认证中的应用是由 Feige,Fiat 和 Shamir 提出的[80,75]。Fiat-Shamir 身份认证方案[80]基于 Goldwasser,Micali 和 Rackoff 提出的二次剩余的零知识证明[124]。

计算合理证明系统(讨论) 计算合理证明系统(即理论)[①]由 Brassard,Chaum 和 Crépeau 提出[40]。他们在同一篇论文中还同时提出了基于分解的困难性的 ***NP*** 完备零知识证明。Naor 等人[171]说明了如何构造基于单向置换的 ***NP*** 完备零知识证明,构造 4.8.3 就选自他们的论文。(4.8.4 小节的)***NP*** 多项式对数通信讨论系统源自 Kilian[143]。

常数轮零知识协议 基于无爪集合的 ***NP*** 轮有效零知识证明系统选自[105]。基于单向函数的 ***NP*** 轮有效零知识讨论源自[77],然而我们的表述(使用了他们的一些思想)有些不同(4.9.2.3 节中描述的可选构造与[77]中的构造更相似)。

非交互零知识证明 非交互零知识证明系统由 Blum,Felman 和 Micali[34]引入。4.10 节中提出的构造出自 Feigh,Lapidot 和 Shamir[76]。欲知关于注释 4.10.6 的更多细节,参见[23]。

多证明者零知识证明 多证明者交互证明由 Ben-Or,Goldwdsser,Kilian 和 WigderSon[26]提出。他们的论文还提出了一个 ***NP*** 完备零知识两证明者证明系统。4.11 节中提出的 ***NP*** 完备零知识两证明者证明就源自他们的思想;然而,我们明确说明了使用的两证明者承诺方案的特性。因此,我们注意到,(足够多次)并行重复这个特定的证明系统将减小错误概率到可以忽略的程度。[②](在 4.11.3 小节末尾简要介绍的有效性改进出自[66]。)

我们指出,多证明者交互证明系统与概率可检验证明(PCP)系统相关。这些证明系统的复杂性理论已经构成了关注的焦点。感关注的读者可以查阅[97]的 2.4 节和 2.5.2 小节(及其参考文献)。

4.12.2 关于进一步阅读的建议

Goldreich 在[97]中介绍了概率证明系统的广阔前景:特别地,[97]的第 2 章包含关于交

① 不幸的是,本文中的术语有些混淆:在某些工作中(特别地,[40]中),计算合理证明(讨论)被粗心地叫做"交互证明。"

② 这里与[146]的作者们不同,由于他们意识到了(通用多证明者系统的)并行重复所带来的问题,因此提出了一种可选构造。

互证明系统的进一步细节，对概率可检验证明(PCP)的介绍以及关于其他类型概率证明系统的讨论。这些介绍把焦点放在了这些系统的基本定义和结果上，并强调了不同类型的概率证明之间的相似性和差别。特别地，像零知识证明系统一样，所有概率证明系统都有一个共同的(非传统)特点：它们都具有一定的错误概率。然而这种错误概率显然是有限的，并且能够通过对证明系统的连续应用来降低。这种非传统的放松所得到的好处是实质性的，就像关于交互证明、零知识证明和概率可检验证明的3个著名的结果所论证的那样：在每种情况中，允许存在有限的错误概率使得这些系统比传统的(无差错的)系统性能更佳，更有用。

自从18年前零知识证明提出以来，零知识证明已经成为许多研究的中心。我们不会提出一份关于进一步阅读的全面建议列表。我们只是指出一些弥补本章中明显漏洞的工作。

- [94]中提供了零知识均匀复杂性的研究。特别地，它给出了如何利用(均匀)单向函数构造不可能找到证明者泄露知识的实例的 ***NP*** 交互证明系统。
- 统计(又叫几乎完备)零知识证明提供了证明者和验证者的绝对安全级别；也就是说，零知识特性和合理性条件从强概率意义上，而不是从计算意义上得到满足。具有统计零知识证明的问题类(记为 ***SZK***)是非常吸引人的(例如，它包含一些困难问题[124,109]，它具有完全问题[194,118,120]，它在求补运算下是封闭的[180,194,118])。感兴趣的读者可参考 Vadhan 的论文[206]。

我们指出，为研究 ***SZK*** 而开发的一些技术也可应用于普通(计算)零知识证明的环境中(例如，从在诚实的验证者的情况下是零知识的公开硬币证明系统到通常意义上就是零知识的类似系统的转换[118])。

- 在4.5节，我们讨论了在零知识环境下并行重复所产生的问题。正如那里提到的那样，并行重复在计算合理性证明[19]环境下和多证明者证明[74,190]环境下也是有问题的。
- 作为4.9节的继续，我们指出，基于离散对数问题的难解性的 ***NP*** 轮有效完备零知识理论已经发表[42]。
- 作为4.10节的继续，我们指出，基于与[76]中相同假设的 ***NP*** 非交互证明系统的一个更有效的构造已经在[144]中出现。[193]中给出了关于非交互零知识进一步强化的建议。
- Goldwasser，Micali 和 Rackoff[124]的论文也包含了对证明者公布的"知识"的通用度量的建议。关于这个叫做知识复杂性的度量的进一步细节，参见[116]及其参考文献(实际上，知识复杂性为0即为零知识)。

最后，我们指出，最近进行的研究是关于在例如同时异步执行[68,189,60]和可重置执行[47]的情况下零知识特性的保持性。在当前阶段企图总结那些研究是不明智的。

4.12.3　未决问题

我们的零知识公式化描述(例如，定义4.3.1中定义的完备零知识)同这门学科使用的标准定义(例如，定义4.3.6)是不同的。标准定义指的是期望多项式时间机器，而不是严格(概率)多项式时间机器。显然，定义4.3.1隐含定义4.3.6(参见习题7)，但是并不清楚反之是否成立。特别地，已知的 ***NP*** 常数轮零知识协议仅当允许期望多项式时间模拟器时才是零知识的。这种情形是令人烦恼的，解决这个问题无论在理论上还是在实践中都有重要意义。

而 **NP** 零知识证明可基于任何(非均匀)单向函数(这是在本书使用的假设中最一般的假设)构造,先前提到的其他一些结果要求更强的假设。特别地,基于比我们当前所用的假设更弱的假设,构造 **NP** 常数轮零知识证明,完备零知识理论和 **NP** 非交互零知识证明会更好。

4.12.4　习题

第一部分的练习是关于基础知识的(即 4.1～4.4 节)。

习题 1:减小交互证明系统的错误概率:证明命题 4.2.7。

解题指导:执行较弱的交互证明足够多次,并为每次执行使用独立选择的硬币投掷,将接受执行的次数与一个合适的门限相比较。注意到完备性和合理性界限需要是可有效计算的。当描述结果验证者的合理性时一定要小心(即,不要假定企图欺骗的证明者独立地执行每个副本)。我们指出,无论这些重复是顺序执行的还是“并行”执行的,这个结论都有效,但是证明顺序重复情况下的合理性条件更容易一些。

习题 2:交互证明中随机化的作用,第 1 部分:证明如果 L 有一个验证者确定的交互证明系统,那么 $L\in$ **NP**。

解题指导:注意到如果验证者是确定的,那么证明者和验证者之间的整个交互可以由证明者确定。

习题 3:交互证明中随机化的作用,第 2 部分:证明如果 L 具有一个交互证明系统,那么 L 就具有证明者确定的证明系统。此外,证明对于每个(概率)交互机器 V,都存在一个确定的交互机器 P,使得对于每个 x,概率 $\Pr[\langle P,V\rangle(x)=1]$ 等于 $\Pr[\langle B,V\rangle(x)=1]$ 在所有交互机 B 上的上确界。

解题指导:对于每个可能的交互前缀,证明者可以确定一个最大化验证者 V 的接受概率的消息。

习题 4:交互证明中随机化的作用,第 3 部分:考虑下面对链接交互机对(和交互证明)的定义的(坏)修改。通过修改,证明者和验证者的随机磁带是一致的(即,直观地讲,他们都使用两者共知的相同硬币投掷序列)。我们称这样的证明系统为共享随机性的交互证明系统。说明只有在 **MA** 中的语言才有共享随机性的交互证明系统,其中语言 L 在 **MA** 中,如果存在一个 **BPP** 中的语言 R_L 和一个多项式 p,使得 $x\in L$ 当且仅当存在 $y\in\{0,1\}^{p(|x|)}$ 使得 $(x,y)\in R_L$。

解题指导:首先将共享随机性的交互证明系统转换为一个验证者公布所有它的硬币投掷的(普通类型的)交互证明系统。然后,使用习题 2 中的推理方法。

说明 **MA** 实际上等于具有共享随机性交互证明系统的语言的集合。

习题 5:交互证明中错误的作用:证明如果 L 具有一个验证者总不接受不在语言 L 中的字符串的交互证明系统,那么 $L\in$ **NP**。

解题指导:定义一个关系 R_L,使得如果 y 是一个导致验证者接受输入 x 的交互的完全副本,那么 $(x,y)\in R_L$。我们强调,y 含有验证者的硬币投掷和所有从证明者那里接收到的消息。

习题 6:完备零知识模拟器中的模拟器错误,第 1 部分:考虑对定义 4.3.1 的修改,在这个修改中,条件 1 被要求对某个函数 $\beta(\cdot)$,$\Pr[M^*(x)=\perp]<\beta(|x|)$ 取代。假设 $\beta(\cdot)$ 是多项

式时间可计算的。说明下列结论成立：

1. 如果对于某个多项式 $p_1(\cdot)$ 和所有足够大的 n，$\beta(n)<1-(1/p_1(n))$，那么这个修改过的定义就等同于原来的定义。

2. 如果对于某个多项式 $p_2(\cdot)$ 和所有足够大的 n，$\beta(n)>2^{-p_2(n)}$，那么这个修改过的定义就等同于原来的定义。

评价函数 $\beta(\cdot)$ 的界限。

解题指导：调用模拟器足够多次。

习题 7：完备零知识模拟器中的模拟器错误，第 2 部分：证明定义 4.3.1 可以导出定义 4.3.6。

习题 8：完备的与几乎完备的零知识：证明每个完备零知识系统也是几乎完备零知识系统（即，证明定义 4.3.1 可以推出定义 4.3.4）。

解题指导：使用习题 6 的第二项，注意 $M^*(x)$ 和 $m^*(x)$（即，"在不输出⊥条件下的 $M^*(x)$"）的统计差别是可以忽略的。

习题 9：计算零知识模拟器中的模拟器错误：考虑定义 4.3.2 的一种可选定义，在这种可选定义中模拟器可以（以概率上限，比如$\frac{1}{2}$）输出符号⊥，并且考虑在它的输出不是⊥条件下的输出分布（同定义 4.3.1 中一样）。证明这种可选定义与原定义（即定义 4.3.2）是等价的。

习题 10：模拟零知识交互的零知识可选公式化描述：证明定义 4.3.2 与定义 4.3.3 的等价性。

解题指导：为了说明定义 4.3.3 可以推导出定义 4.3.2，注意到每个交互机的输出可以很容易地从它对交互的观点中计算出来。为了说明定义 4.3.2 可以推导出定义 4.3.3，只要说明对于每个概率多项式时间交互机 V^*，都存在一个概率多项式时间交互机 V^{**}，使得 $\text{view}_{V^*}^{P}(x)=\langle P,V^{**}\rangle(x)$。

习题 11：证明定义 4.3.10 等价于一个新的定义，在这个新的定义中，验证者的辅助输入具有明显的长度限制。换句话说，这个新的定义可如下表述。

对于每个多项式 l 和每个概率多项式时间交互机 V^*，都存在一个概率多项式时间算法 M^*，使得下面的两个总体是计算不可分辨的：

- $\{\langle P(y_x),V^*(z)\rangle(x)\}_{x\in L,z\in\{0,1\}^{l(|x|)}}$
- $\{M^*(x,z)\}_{x\in L,z\in\{0,1\}^{l(|x|)}}$

其中 y_x 与定义 4.3.10 中一样。

注意，这里 M^* 的运行时间（和分辨距离）被看作是 $|x|$ 的函数还是 $|x,z|$ 的函数并不重要。

习题 12：给出一个简单概率多项式时间算法，这个算法可模拟构造 4.3.8 中的验证者和那里定义的证明者交互的观点。对于输入 $x\in GI$，这个模拟器应该能够产生与 $\text{view}_{V_{GI}}^{P_{GI}}(x)$ 同分布的输出。

习题 13：证明比特承诺方案的存在性可以推导出单向函数的存在性。

解题指导：遵循定义 4.4.1 中的符号，考虑 (v,s,r) 到接收者观点 $(r,\overline{m})$ 的映射。注意到根据明确性要求，定义域内的元素不可能有两个可能 v 值的逆。这个映射是多项式时间可计算的，任何以不可忽略的成功概率求逆的算法都与保密性要求相矛盾。

习题 14：考虑构造 4.4.4 中的承诺方案，提出一个试图欺骗的发送者，这个发送者可以推

导出(承诺阶段的)接收者观点,并且这个推导出的接收者观点不可能有任何可能的公布,而且与接收者对与指定的发送者交互的观点是计算不可分辨的。即,给出一个概率多项式时间交互机 S^*,满足下面两个条件:

1. 以非常高的概率,$\langle S^*(0),R\rangle(1^n)$既不是一个可能的 0 承诺,也不是一个可能的 1 承诺。

2. 总体$\langle S^*(0),R\rangle(1^n)$和$\langle S(0),R\rangle(1^n)$是计算不可分辨的。

解题指导:发送者简单地以一个均匀选择的字符串来答复就可以。

习题 15:将构造 4.4.4 作为构造 4.4.7 中的承诺方案:证明当构造 4.4.4 中的承诺方案在 $G3C$ 协议中使用时,得到的结果方案保持零知识特性。考虑需要进行的修改来证明结论 4.4.8.2。

习题 16:强约简:令 L_1 和 L_2 是 **NP** 中的两个语言,并令 R_1 和 R_2 是分别刻划 L_1 和 L_2 的二元关系。我们说关系 R_1 是关系 R_2 的 Levin 约简[①],如果存在两个多项式时间可计算函数 f 和 g,满足下面两个条件:

标准要求:$x\in L_1$ 当且仅当 $f(x)\in L_2$。

附加要求:对于每个$(x,w)\in R_1$,都有$(f(x),g(x,w))\in R_2$。

证明下列结论:

1. 令 $L\in$ **NP**,并令 R_L 是刻划 L 的关系(即,给定一个非确定性机器 M_L,如果 w 是M_L 在输入 x 上的可接受计算,令$(x,w)\in R_L$)。令 R_{SAT}是刻划SAT 的标准关系(即,如果 w 是一个满足 CNF 公式 x 的真值分配,则有$(x,w)\in R_{SAT}$)。证明 R_L 可 Levin 约简为关系 R_{SAT}。

2. 令 R_{SAT}同上,令 R_{3SAT}是对 $3SAT$ 的类似定义。证明 R_{SAT}可 Levin 约简为关系 R_{3SAT}。

3. 令 R_{3SAT}同上,令 R_{G3C}是刻划 $G3C$ 的标准关系(即,如果 w 是图 x 的三着色,就有$(x,w)\in R_{G3C}$)。证明 R_{3SAT}可 Levin 约简为关系 R_{G3C}。

4. Levin 约简具有传递性。

习题 17:证明当 SAT 被看作是一个可以在多项式时间内求逆的函数时,存在一个从任何 **NP** 语言 L 到 SAT 的 Karp 约简。对于 SAT 到 $3SAT$ 的约简和 $3SAT$ 到 $G3C$ 的约简也同样成立(实际上,标准 Karp 约简具有这个特性)。

习题 18:定理 4.4.11 的应用:这个习题假设读者基本熟悉公钥加密方案和签名方案的概念。假设存在非均匀单向函数,给出如下密码问题的解决方案。

1. 假设称为 S 的一方在公共信道上将加密的数据发送给多方 $R_1,\ldots,R_t$。特别地,发送给 R_i 的数据用R_i 的公钥进行加密。我们假设所有参与方都可以访问在公共信道上传输的密文。假设 S 想要向另一方证明它已经将相同的数据发送给了所有的R_i,但是又不想在证明的同时泄漏数据。

2. 在同样的通信环境中,考虑已经接收到用它自己的公钥加密的数据的一方 R。假设这些数据由两部分组成,并且 R 希望向某人公布数据的第一部分但是不想公布数据的第二部分。进而假设另一方想要一个证据来说明 R 确实公布了数据第一部分的正确内容。

---

① 我们用 Levin 来命名这个约简,是因为当 Levin(与 Cook 和 Karp 独立地)发现 **NP** 完全问题的存在性时,使用了一个更强的可以推导出这里使用的约简定义的约简定义。我们假设读者对诸如 Bounded Halting、SAT 和 3SAT(同[86])等语言中的标准约简很熟悉。

3. 假设 S 希望将一个众所周知文档的签名发送给 R，使得只有 R 才能接收这个签名，但是其他任何人都可以验证这个签名确实是 S 发送的（我们再一次假设所有参与方都公用一个公共信道）。

习题 19：知识紧密性：证明并行执行构造 4.4.7 $k(n)=O(\log n)$ 次得到的协议是零知识，并证明它具有知识紧密性 $(3/2)^{k(n)}$（近似）。

习题 20：更有效的 ***NP*** 零知识证明：考虑构造 4.7.14 中给出的哈密尔顿环路问题（HC）的基本证明系统。

1. 估计它的接受概率（即，完备性和合理性界）。

2. 提供一个零知识特性证明的梗概（即，描述一个模拟器）。特别地，给出一个建立知识紧密性大约为 2 的模拟器。如果有兴趣，给出零知识特性的一个完整证明。

高级章节的练习 剩下的练习与高级章节（即 4.5～4.11 节）的内容有关。

习题 21：黑盒零知识的一个可选公式化描述：这里我们说一个概率多项式时间预言机 M 是一个对于证明者 P 和语言 L 的**黑盒模拟器**，如果对于每个（不一定是均匀的）多项式规模电路族 $\{B_n\}_{n\in\mathbb{N}}$，总体 $\{\langle P,B_{|x|}\rangle(x)\}_{x\in L}$ 和总体 $\{M^{B_{|x|}}(x)\}_{x\in L}$ 对于（非均匀）多项式规模电路是不可分辨的。即，对于每个多项式规模电路族 $\{D_n\}_{n\in\mathbb{N}}$、每个多项式 $p(\cdot)$ 以及所有足够大的 n 和 $x\in\{0,1\}^n\cap L$，都有

$$|\Pr[D_n(\langle P,B_n\rangle(x))=1]-\Pr[D_n(M^{B_n}(x))=1]|<\frac{1}{p(n)}$$

证明现在的公式化描述与定义 4.5.10 中的等价。

习题 22：证明构造 4.4.7 中给出的协议实际上是一个 $G3C$ 的黑盒零知识证明系统。

解题指导：使用习题 21 中的公式化描述。

习题 23：证明黑盒零知识在顺序合成下保持不变（注意，这不能仅仅通过辅助输入零知识在顺序合成下保持不变的事实得出）。

解题指导：采用引理 4.3.11 的证明。

习题 24：反驳另一个并行合成猜想：证明存在一个零知识证明者 P，使得并行运行两个 P 的副本得到的证明者会泄漏知识（例如，一个试图欺骗的验证者可以从这个证明者那里得到一个在多项式时间里不可能解决的问题的解）。

解题指导：令 P_1 和 P_2 同命题 4.5.9，考虑随机选择两个程序中的一个进行执行的证明者 P。这个选择也可以由验证者来决定。

习题 25：假设存在单向置换，给出一个不是强证据不可分辨的（具有一个概率多项式时间证明者的）证据不可分证明系统。

解题指导：考虑一个单向置换 f，f 的一个核心断言 b 和证据关系 $\{(f(w),w):w\in\{0,1\}^*\}$。考虑在输入为 $f(w)$（和辅助输入为 w）时发送 w 给验证者的证明者，并考虑总体 $\{X_n^0\}_{n\in\mathbb{N}}$ 和总体 $\{X_n^1\}_{n\in\mathbb{N}}$，其中 X_n^i 在 $\{f(w):w\in\{0,1\}^n\ \&\ b(w)=i\}$ 上服从均匀分布。

习题 26：一些知识的基本零知识证明：

1. 说明构造 4.3.8 是一个知识错误为 $\frac{1}{2}$ 的同构知识的证明。

2. 说明构造 4.4.7（当应用在公共输入 $G=(V,E)$ 上时）是一个知识错误为 $1-\frac{1}{|E|}$ 的三着色知识的证明。

解题指导：注意到在这些情况下，如果验证者以大于知识错误的概率接受，那么它就以概率 1 接受。还注意到这些证明系统中可能的验证者消息数是公共输入的多项式。因此，提取者可以模拟具有所有可能验证者消息的证明系统的执行。

习题 27：某些基本知识证明的并行重复：令 $k:\mathbb{N}\rightarrow\mathbb{N}$ 是多项式受限的。考虑通过 k 次并行执行习题 26 中提到的基本系统得到的证明系统。

1. 说明构造 4.3.8 的 k 次并行执行构成一个具有知识错误 $2^{-k(\cdot)}$ 的同构知识的证明（类似的结论对构造 4.7.12 也成立）。

2. 说明构造 4.4.7 的 k 次并行执行会得到一个具有知识错误 $(1-(1/|E|))^{-k(|G|)}$ 的三着色知识的证明。

注意，我们没有任何关于零知识的声明。

也可以参考习题 28 的第 2 部分。

解题指导：对于第 1 部分，注意到任何两个验证者接受的不同原证明的执行副本都会产生一个同构。在第 2 部分中，这一点不成立。还会注意到，与任何基本系统的固定副本不同 $|E|$ 个验证者接受的副本确实会产生一个三着色。

习题 28：***NP*** 知识的更有效的零知识证明：同习题 20，考虑构造 4.7.14 中给出的关于哈密尔顿环路问题（HC）的基本证明系统。

1. 证明这个基本证明系统是一个知识错误为 $\frac{1}{2}$ 的哈密尔顿环路问题的知识证明。

2. 证明通过 k 次迭代基本系统得到的证明系统是一个知识错误为 2^{-k} 的哈密尔顿环路的知识证明。要考虑顺序重复和并行重复两种情况。

习题 29：定义 4.7.2 和定义 4.7.3 的等价性的进一步问题：假设 R 是多项式受限的，并且定义 4.7.3 中的提取器或者输出一个有效解或者输出一个特定的失败符号。关于这个关系 R，说明 V 满足定义 4.7.2 中具有错误 κ 的有效性条件当且仅当 V 满足（修改的）定义 4.7.3 中具有错误 κ 的有效性条件。

解题指导：遵循命题 4.7.4 的证明要点，注意到 R 是 ***NP*** 关系的假设可以用定义 4.7.3 中的提取器或者输出有效解或者输出一个特定的失败符号的假设来代替。特别地，在充分性证明中，省略以概率 $2^{-\text{poly}(|x|)}$ 发生的穷搜索，并运用 $p(x,y,r)>\kappa(|x|)$ 可以推出 $p(x,y,r)\geq\kappa(|x|)+2^{-\text{poly}(|x|)}$ 这一事实。

习题 30：***NP*** 知识的零知识强证明：再次考虑构造 4.7.14 中提出的哈密尔顿环路问题（HC）的基本证明系统。证明通过足够多次顺序迭代基本系统得到的证明系统是一个哈密尔顿环路知识的强证明（实际上它是零知识的）。

习题 31：在计算合理证明中的错误约简：给定一个语言 L 的（具有错误概率 $\frac{1}{3}$ 的）计算合理证明，构造一个（L 的）具有可忽略错误概率的计算合理证明。

解题指导：运用顺序重复。实际上，错误概率可以指数减小。在某些情况下，并行重复可能不会减小计算合理性（见[19]）。

习题 32：承诺方案，一个不可能的结果：证明不存在同时满足定义 4.8.2 中的完备保密性要求和定义 4.4.1 中的（信息理论）明确性要求的双方协议。

习题 33：构造 4.8.3 中普通哈希的失效：说明在构造 4.8.3 中，将迭代哈希替换为一个普通的哈希会得到一个不具有（甚至是计算意义上的）绑定性的方案。也就是说，使用构造 4.8.3 的概念，考虑用如下步骤代替迭代哈希的步骤（其中 b 和 r^i 与构造 4.8.3 中的相同）。

- （普通哈希）：接收者发送消息$(r^1,\ldots,r^{n-1})$给发送者，发送者用消息$(c^1,\ldots,c^{n-1})$来应答，其中对于 $i=1,\cdots,n-1$，$c^i \stackrel{\text{def}}{=} b(y,r^i)$。

也就是说，指定的发送者像构造 4.8.3 中那样计算 c^i，但是企图欺骗的发送者可以根据所有的 r^i 来决定所有的 c^i（而不是仅根据$(r^1,\ldots,r^i)$来决定每个 c^i）。

提出一个有效的策略，允许发送者违反明确性条件。

解题指导：给定任意单向置换 f'，首先构建一个单向置换 f，满足对于每个 x' 有 $f(0^{|x'|},x')=(0^{|x'|},x')$和 $f(x',0^{|x'|})=(x',0^{|x'|})$。（提示：首先得到一个单向置换 f''，满足对于所有的 n[①] 有 $f''(0^n)=0^n$，然后对于 $x',x''\in\{0,1\}^{|x'|}\setminus\{0\}^{|x'|}$，令 $f(0^{|x''|},x'')\stackrel{\text{def}}{=}(0^{|x''|},x'')$，$f(x',0^{|x'|})\stackrel{\text{def}}{=}(x',0^{|x'|})$和 $f(x',x'')\stackrel{\text{def}}{=}(f''(x'),f''(x''))$。）

假设修改的协议用这里构造的 f 执行，考虑一个试图欺骗的发送者，当接收到消息$(r^1,\ldots,r^{n-1})$时，这个发送者找到 $y^1\in\{0,1\}^{n/2}\{0\}^{n/2}$，$y^2\in\{0\}^{n/2}\{0,1\}^{n/2}$ 和 $\bar{c}=(c^1,\ldots,c^{n-1})$，满足如下条件：

对于 $i=1,\ldots,n-1$ 和 $j=1,2$，有 $c^i=b(y^j,r^i)$。

对于 $j=1,2$，$b(y^j,r^n)\equiv j(\text{mod}2)$。

（其中 r^n 是惟一的与 $r^1,\ldots,r^{n-1}$独立的向量。）

注意到，f 在这样的 y^j 下是不变的，因此它们可以作为有效的解除承诺。

最后，证明这样的解 y^1,y^2 和$\bar{c}$总存在，并可以通过解一个线性系统得到。（提示：对于 $i=1,\ldots,n-1$ 和 $b(x^1 0^{n/2} x^2,r^n)+1(\text{mod}2)$，考虑线性系统 $b(x^1 0^{n/2},r^i)=b(0^{n/2}x^2,r^i)=b(0^{n/2}x^2,r^i)$。其他提示：当将条件写成矩阵形式时，就会更加明显。）

习题 34：非交互零知识，有约束的与无约束的：说明除非 ***NP***$\subseteq$***BPP***，否则构造 4.10.4 不是无约束的零知识。

解题指导：考虑调用这个证明系统两次：第一次对包含一个简单环路的图调用，然后对包含一个哈密尔顿环路的图调用。

习题 35：关于有两个发送者的承诺方案的定义（定义 4.11.4），说明对于每个 p，都存在发送者的策略，使得每个得到的接收者观点都可以以概率 p0 公开，以概率 $1-p$1 公开。

解题指导：使用完备保密性要求和你可以给出计算无约束发送者策略的事实。

---

① 见第 2 章的习题 13。

附录 A 计算数论背景

此附录介绍的内容只能满足本书介绍的几个具有特殊结构的例子的最低需要，主要是几个有关素数和合数的结构与算法的问题。要想获得更深刻的理解，可进一步参考标准的教科书（如：[10]）。

A.1 素　　数

素数即除了 1 和它本身外再没有其他因子的自然数。为了简单起见，规定 1 不是素数。

对一个素数 P，模 P 加群记为 $\mathbb{Z}_P$，它由集合 $\{0,\ldots,P-1\}$ 和定义在其上的模 P 加法运算构成。在 $\mathbb{Z}_P$ 中除 0 元外，所有元素的阶都是 P。模 P 乘群记为 $\mathbb{Z}_P^*$，它由集合 $\{0,1,\ldots,P-1\}$ 和定义在其上的模 P 乘法运算构成。$\mathbb{Z}_P^*$ 是一个循环群。事实上，群中至少有 $1/\log_2 P$ 个元素的阶是 $P-1$，我们把这些元素称为本原元。①

A.1.1 素数求模的二次剩余

设 P 是一个素数，模 P 的二次剩余是一个整数，不妨记为 s，如果存在一个 $r\in\mathbb{Z}_P^*$，满足 $s\equiv r^2 \pmod P$，则称 s 是模 P 的二次剩余。因此特别指出，s 与 P 必须互素。显然，如果 r 是 s 的模 P 平方根，则 $-r$ 也是（因为 $(-r)^2\equiv r^2$）。进一步，如果方程 $x^2\equiv s \pmod P$ 在 $\mathbb{Z}_P$ 中有一个解，那么它肯定恰有两个这样的解（否则，设 $r_1\not\equiv\pm r_2 \pmod P$ 都是解，从而 $0\equiv r_1^2-r_2^2\equiv(r_1-r_2)(r_1+r_2) \pmod P$，与 P 是素数相矛盾）。

模 P 的二次剩余构成了模 P 乘群的子群。这样形成的子群恰好包含群的一半元素。而且，平方模 P 运算正好是群到子群的 2-1 的映射。在 $P\equiv 3 \pmod 4$ 的情况下，这个映射的每一个像在子群中有一个原像（即一个二次剩余），另一个原像不在子群中（即一个非二次剩余）。②

A.1.2 在素数求模运算中开方

一般来讲，可以用 Berlekamp 算法在素数求模运算中开方[28]。它是一个素数求模运算

① 模 P 本原元的个数取决于素因子分解 $P-1=\prod_{i=1}^{t} P_i^{e_i}$，（见 A.2 节）；实际上等于 $\prod_{i=1}^{t}((P_i-1)\cdot P_i^{e_i-1})$。

② 因为 -1 是模这些素数的非二次剩余。相比之下，如果 $P\equiv 1 \pmod 4$，-1 就是模 P 的一个二次剩余。因此，当 $P\equiv 1 \pmod 4$ 时，对每一个二次剩余，其两个平方根要么都是二次剩余，要么都是非二次剩余。

中分解多项式的随机算法(因为求 s 模 P 的平方根实际上是解方程 $x^2 \equiv s(\mathrm{mod}\ P)$,从而可通过分解多项式 $x^2 - s(\mathrm{mod}\ P)$来计算)。

如果对应的素数是与 3(*mod* 4)同余的,这是在大多数密码学应用中的情况,在这种情况下将有一个更直接的方法。对一个二次剩余 $s \equiv x^2(\mathrm{mod}\ P)$,我们有

$$\begin{aligned} s^{(P+1)/4} &\equiv x^{(P+1)/2}(\mathrm{mod}\ P) \\ &\equiv x^{(P-1)/2} \cdot x\ (\mathrm{mod}\ P) \\ &\equiv \pm x\ (\mathrm{mod}\ P) \end{aligned}$$

在最后一个等式中我们用了 Fermat 小定理,由它我们知道,对每一个整数 x 和素数 P,有 $x^{(P-1)/2} \equiv \pm x(\mathrm{mod}\ P)$。这样,在这种特殊的情况下,通过求 $s^{\frac{P+1}{4}}(\mathrm{mod}\ P)$,就得到了 s 的模 P 平方根(注意,这个平方根是模 P 的一个二次剩余)。

A.1.3 素性检测器

检测一个整数是否为素数的一般方法是利用 Rabin 的素性检测器[185],这种检测器与 Miller[166]的确定性算法相联系。[1] 还有一种稍微有些不同的随机算法,分别由 Solovay 和 Strassen[202]独立发现,也可以用来检测,但似乎不太普及。这里介绍第三种可选的检测器,它看起来似乎更不出名(由几位研究人员分别发现,其中一位是 Manuel Blum)。我们仅仅使用了如下数论事实:

1. 对每一个素数 $P>2$,每一个模 P 的二次剩余都恰有两个模 P 的平方根(且它们的和为 P)。

2. 对每一个奇的非整数次幂的合数 N,每一个模 N 的二次剩余至少有 4 个模 N 的平方根。

我们使用的是黑箱算法,记为 SQRT,给定素数 P 和模 P 的二次剩余 s,返回 s 模 P 的平方根。但是不能保证如果输人不是这种形式时,算法将如何运行(特别是当 P 不是素数的时候)。

算法 输入一个自然数 $N>2$,具体步骤如下:

1. 如果 N 是一个偶数,或者是一个整数的幂次,就拒绝。
2. 均匀选择 $r \in \{1, \ldots, N-1\}$,并令 $s \leftarrow r^2 \bmod N$。
3. 令 $r' \leftarrow \mathrm{SQRT}(N, s)$,如果 $r' \equiv \pm r\ (\mathrm{mod}\ N)$,就接受,否则就拒绝。

分析 由事实 1,输入一个素数 N,算法常常接受(因为此时,对任意 $r \in \{1, \ldots, N-1\}$,都有 $\mathrm{SQRT}(N, r^2 \bmod N) = \pm r$。另一方面,如果 N 是一个奇合数,且不是一个整数的幂,那么由事实 2,每一个二次剩余 s 都至少有 4 个平方根,每一个都等可能地在步骤 2 中被选中(因为对每个特定的 r,s 都没有隐含更多的信息)。这样,对每一个这样的 s,在步骤 2 中 $\pm\mathrm{SQRT}(N, s)$被选中的概率至多为 $\frac{2}{4}$。从而,输人合数时算法至少以 $\frac{1}{2}$ 的概率拒绝。

注释 分析预设算法 SQRT 在输入为(P,s)时总是正确的,这里 P 是一个素数,s 是模 P 的二次剩余。我们在特殊情况 $P \equiv 3(\mathrm{mod}\ 4)$下描述了这样的算法。因此,只要选择的数是与

[1] Miller 的算法的依据是推广的黎曼假设(ERH)。

3(mod4)同余的，也正好是我们的应用中的情况，描述就是正确的。当 $P\equiv 1 \pmod 4$ 时，应用前面提到的随机求模开方算法，我们发现当 SQRT 的错误概率 $\varepsilon<\frac{1}{2}$ 时，算法仍能分辨素数和合数(因为对于素数，我们至少以 $1-\varepsilon>\frac{1}{2}$ 的概率接受，而对于合数，我们至多以 $\frac{1}{2}$ 的概率接受)。调用算法多次可扩大两种情况下的统计差异，从而使结果更精确。

我们指出检测素性的无错误的不确定性多项式时间算法确实存在[121,1]，但目前运行非常缓慢(这种算法要么输出正确答案，要么输出一个特殊的未知符号，且输出后者的概率至多为 $\frac{1}{2}$)。

A.1.4 素数的均匀选择

在某个区间内均匀生成一个素数的简单方法是，在这个区间重复地随机选择一个整数，并检测它的素性，不妨设区间为 $[N,2N]$。当然，问题是要找到一个素数，我们需要重复多少次算法。这个问题与素数的稠密度密切相关，素数的稠密度问题是数论中广泛研究的一类问题[7]。为达到我们的目的，可以假设抽样空间足够大(与它内部的整数长度相比)，从而素数的密度可以很明显地计算(即，是一个多项式分数)。特别地，区间 $[N,2N]$ 的素数密度为 $\Theta(1/\log N)$。因此，当输入为 N 时，在 $[N,2N]$ 内找到一个素数平均要试验 $\Theta(\log N)$ 次。进一步，在试验次数少于 $\Theta((\log N)^2)$ 时，我们至少能以 $1-(1/N)^2$ 的概率得到一个整数。因此，在所有的实际问题中，我们只需进行一定数量的试验，具体可能是我们要得到的素数的长度的一个多项式(即 n^2)(对于与 3 mod 4 同余的素数有类似的讨论)。

注意到存在一个不确定多项式时间算法[9]，它能够生成均匀选择的一个素数 P，同时得到 $P-1$ 的一个因式分解。$P-1$ 的素因子分解可以验证一个给定的剩余是否为模 P 乘群的一个生成元：如果 $g^{P-1}\equiv 1 \pmod P$，且对每一个整除 $P-1$ 的 N，$g^N \not\equiv 1 \pmod P$，则 g 为 $\mathbb{Z}_P^*$ 的一个生成元。(对每一个能整除 $P-1$ 的素数 Q，都能够检测 $g^{(P-1)}\equiv 1 \pmod P$ 和 $g^{(P-1)/Q}\not\equiv 1 \pmod P$。)我们指出模 P 剩余类的值得注意的一部分可能是 $\mathbb{Z}_P^*$ 的生成元。

最后，我们指出只利用 $O(n)$ 个随机位生成 n 位长的素数的更随机有效的算法确实存在。①

A.2 合　　数

我们把自然数(1 除外)中不是素数的数称为合数。每一个合数 N 都可惟一地表示为几个素数次幂的乘积；也即，$N=\prod_{i=1}^{t} P_i^{e_i}$，这里 P_i 是互不相同的素数，e_i 是自然数，且 $t>1$ 和 $e_1>1$ 至少有一个成立。这些 P_i 称为 N 的素因子。目前普遍认为，给定一个合数，求它的素因子分解几乎是不可行的。特别地，我们假设给定一个由两个随机素数相乘得到的合数，再把它分解为两个素数的乘积是不可行的。也即，我们假设给定两个均匀选择的 n 位长的素数的乘积，任意的非确定多项式时间算法将它们还原的概率几乎都可以忽略。Rivest，Shamir 和 Adle-

① 例如，可以用[177]的一个一般的变形。这个变形将任意多项式时间线性空间随机算法变为一个具有线性随机复杂性的类似算法。前文描述的选择过程必须满足变形的前提条件。

man[191]建议可以利用这个假设来构造密码算法。事实上他们确实提出了 RSA 函数，并且他们的建议产生了巨大的影响(即，成了密码学应用方面最流行的难处理假设)。

对于一个合数 N，模 N 加群记为 $\mathbb{Z}_N$，它由集合 $\{0,...,N-1\}$ 和定义在其上的模 N 加法运算构成。在 $\mathbb{Z}_N$ 中所有与 N 互素的元素的阶均为 N。模 N 乘群记为 $\mathbb{Z}_N^*$，其中包含的元素是比 N 小且与 N 互素的自然数，该运算是乘法模 N。

A.2.1 合数求模的二次剩余

为了简单起见，我们集中精力考虑奇合数，即不可能分为任何严格素数幂的乘积的合数；也即这些合数具有形式 $\prod_{i=1}^{t} P_i$，这里 P_i 是互不相同的奇素数，且 $t>1$。

设 $N=\prod_{i=1}^{t} P_i$ 为一奇合数。s 是一个整数，如果存在 $r\in\mathbb{Z}_N^*$，满足 $s\equiv r^2 \pmod N$，则称 s 为模 N 二次剩余。利用中国剩余定理，可以得出 s 是模 N 二次剩余当且仅当它是对每一个 P_i 的模 P_i 二次剩余。假设 s 是模 N 二次剩余，则方程 $x^2\equiv s \pmod N$ 有 2^t 个不同的模 N 整数解。可以用中国剩余定理证明这个结论：首先找系统

$$x^2 \equiv s \pmod{P_i} \qquad i=1,...,t \tag{A.1}$$

的一个解。然后注意到每一个方程有两个互不相同的解 $\pm r_i \pmod{P_i}$，最后我们指出对于 2^t 个不同的解的每一个组合都是方程 A.1 的模 N 解(即，不同的 s 的模 N 平方根)。

模 N 的二次剩余构成 Z_N^* 的一个子群。子群元素恰好是群元素的 2^{-t} 倍。此外，对 $N=\prod_{i=1}^{t} P_i$，平方模 N 是一个 2^t-1 的群到子群的映射，对于 $t=2$ 和 $P_1\equiv P_2\equiv 3 \pmod 4$ 时的情况，我们将对这个映射做进一步讨论，详见 A.2.4 节。

A.2.2 合数求模的开方运算

根据前述讨论(和中国剩余定理的有效性)，[①]给定 N 的素因子分解，就能够有效地计算模 N 的平方根。另一方面，任何合数求模的开方算法都可以转化为因式分解算法[187]：能够指出一个开方算法(模 N)如何产生 N 的非平凡因子。这个结论的论据与 A.1.3 小节所用到的非常类似，惟一的不同是那里的开方算法仅仅适用于对素数求模运算，而这里我们假设算法是对合数求模开方的(并假设这样有效的算法是不存在的)。

由因式分解到开方运算的转化 输入一个合数 N，具体步骤如下：

1. 均匀选择 $r\in\{1,...,N-1\}$。
2. 计算 $g\leftarrow \mathrm{GCD}(N,r)$，如果 $g>1$，则输出 g 并终止算法。[②]
3. 令 $s\leftarrow r^2 \bmod N$，并调用开方算法获得 r'，使得 $(r')^2\equiv s \pmod N$。
4. 计算 $g\leftarrow \mathrm{GCD}(N,r-r')$，如果 $g>1$，输出 g 并终止。

当算法有输出并终止时，输出的肯定是 N 的非平凡因子。对 N 的两个非平凡因子分别调用算法多次，就可以得到 N 的素因子分解。

---

① 特别地，对于 $i=1,\cdots,t$，系统 $x\equiv a_i \pmod{P_i}$ 的解为 $\sum_{i=1}^{t} c_i\cdot a_i \bmod \prod_{i=1}^{t} P_i$，其中 $c_i\stackrel{\text{def}}{=}Q_i(Q_i^{-1} \bmod P_i)$，$Q_i\stackrel{\text{def}}{=}\prod_{j\neq i} P_j$。

② 这一步是为了保证只对互素的数对 (s,N) 调用开方算法。

分析　我们可以假设步骤 1 中选取的 r 与 N 互素，否则 r 与 N 的最大公约数就是所要的因子。调用求平方根算法，可得到 r'，使得 $(r')^2\equiv s\equiv r^2 \pmod N$。因为求平方根算法没有隐含 r 的任何信息（除了以 $2/2^t$ 的概率获得 $r^2 \pmod N$），我们有 $r'\not\equiv\pm r \pmod N$。否则，$r'\neq\pm r \pmod N$，从而有 $0=(r-r')(r+r') \pmod N$。因此，$r-r'$（或者，$r+r'$）与 N 有非平凡最大公约数，这个最大公约数将在步骤 4 中找到。从而，我们至少可以以 $\frac{1}{2}$ 的概率获得 N 的非平凡因子。

A.2.3　勒让德和雅克比符号

设 P 是一个素数，整数 r 模 P 的勒让德符号记为 $\mathrm{LS}_P(r)$，定义如下：$\mathrm{LS}_P(r)=0$（r 整除 P）；$\mathrm{LS}_P(r)=+1$（r 是模 P 的二次剩余）；$\mathrm{LS}_P(r)=-1$（其他）。这样，对于与 P 互素的 r，r 模 P 的勒让德符号就表示 r 是否为一个二次剩余。

设 N 是一个合数，模 N 剩余类的雅克比符号的定义是建立在 N 的素因子分解基础上的。设 $N=\prod_{i=1}^{t}P_i^{e_i}$，则 r 模 N 的雅克比符号记为 $\mathrm{JS}_N(r)$，定义为 $\prod_{i=1}^{t}\mathrm{LS}_{P_i}(r)^{e_i}$。尽管 r 模 N 的雅克比符号是按照模数的素因子分解来定义的，但却可以在没有模数的因子分解知识的情况下有效地计算它。也即，存在多项式时间算法，给定数对 (r,N)，就可以计算 $\mathrm{JS}_N(r)$。算法运行和求最大公约数的方式类似，并应用如下有关雅克比符号的事实：

1. $\mathrm{JS}_N(r)=\mathrm{JS}_N(r \bmod N)$
2. $\mathrm{JS}_N(a\cdot b)=\mathrm{JS}_N(a)\cdot\mathrm{JS}_N(b)$，$\mathrm{JS}_N(1)=1$
3. $\mathrm{JS}_N(2)=(-1)^{(N^2-1)/8}$（即，$\mathrm{JS}_N(2)=-1$ iff $N\equiv4\pm1 \pmod 8$）
4. 对于奇整数 N 和 r，$\mathrm{JS}_N(r)=(-1)^{(N-1)(r-1)/4}\cdot\mathrm{JS}_r(N)$

注意到一个模 N 二次剩余的雅克比记号必为 1，但并不是所有雅克比符号为 1 的剩余都是模 N 二次剩余（事实上，对于 A.2.1 小节中的 $N=\prod_{i=1}^{t}P_i$，雅克比符号非 0 的剩余有一半的雅克比记号为 1，但只有 2^{-t} 的剩余是某数的平方模 N）②。可以有效计算雅克比符号（不知道模数的因子分解）这一事实并不能产生一个有效的算法来确定一个给定的剩余是否为一个对给定合数（不知道因子分解）求模的平方数。事实上，现在认为确定一个给定整数是否为对一个给定合数求模的二次剩余是不可行的。Goldwasser 和 Micali[123]提出对这个问题的难处理假设应用到构造密码算法，建议提出后，大量的工作都是根据它来开展的。

A.2.4　布卢姆整数和它们的二次剩余结构

设 $N=P\cdot Q$，这里 P 和 Q 是两个素数，如果 $P\equiv Q\equiv3 \pmod 4$，我们就称 N 为一个布卢姆整数。对这样的 P（或 Q），-1 不是模 P（或 Q）的二次剩余，从而 -1 不是模 N 的二次剩余，-1 的雅克比记号为 1 模 N。

通过前面的讨论，每一个模 N 的二次剩余 s 有 4 个平方根，记为 $\pm x$，$\pm y$，因此 GCD(N，x

---

① 例如 $\mathrm{JS}_{21}(10)=\mathrm{JS}_{21}(2)\cdot\mathrm{JS}_{21}(5)=(-1)^{55}\cdot(-1)^{20}\cdot\mathrm{JS}_5(21)=-\mathrm{JS}_5(1)=-1$。通常，事实 2 只用于 $a=2$（即 $\mathrm{JS}_N(2\cdot r)=\mathrm{JS}_N(2)\cdot\mathrm{JS}_N(r)$）的情形。且在一开始，还可以使用 $\mathrm{JS}_{2N}(r)=\mathrm{JS}_2(r)\cdot\mathrm{JS}_N(r)=(r \bmod 2)\cdot\mathrm{JS}_N(r)$。

② Z_N^* 中雅克比符号为 1 的元素构成它的一个子群。这个子群的元素恰好是群元素的一半。

$\pm y)\in\{P,Q\}$。有关布卢姆整数的一个重要的事实是，4 个之中恰有一个平方根是它自己的二次剩余。[1] 因此，导出模 N 二次剩余集合上的一个置换。

（有些资料采用一个更一般的布卢姆整数的定义，但上述特殊情况已能满足我们的要求。“布卢姆整数”是为了纪念 Manuel Blum 而命名的，他提倡将对这样的数求模的平方运算作为一种单向置换。）

我们指出当 $P\not\equiv Q \pmod 8$ 时，4 模 N 的雅克比符号为 -1 ($N=P\cdot Q$)。在这种情况下，得到 4 mod N 的一个平方根，且它又是自身的二次剩余，就可以对 N 进行分解（因为这样的剩余 r 满足 $r\not\equiv\pm 2 \pmod N$，$(r-2)\cdot(r+2)\equiv 0 \pmod N$）。

---

[1] 令 a 和 b 满足 $a^2\equiv s \pmod P$，$b^2\equiv s \pmod Q$，则 a 和 $-a$ 中有且只有一个是模 P 的二次剩余，对 b 有相同的结论。不失一般性，我们假设 a（或 b）是模 P（或 Q）二次剩余。满足 $x\equiv a \pmod P$ 和 $x\equiv b \pmod Q$ 的 x 是 s 模 N 的一个平方根，且是它的一个二次剩余。S 模 N 的其他平方根（即满足 $y\equiv a \pmod P$ 和 $y\equiv -b \pmod Q$ 的 $-x,\pm y$,）不是模 N 的二次剩余。

第 2 卷摘要

附录 B

本书，也即第 1 卷只介绍了现代密码学的一些基础工具，即单向函数、伪随机性和零知识证明。这些基础工具主要用于基本应用程序（第 2 册中介绍）的构造。这些应用程序包括加密、签名和一般的密码协议。在附录 B 中，我们介绍对这些基本应用程序处理的摘要。

B.1 加密：摘要

不管是私钥加密算法还是公钥加密算法都包含 3 个基本算法：密钥的产生、加密和解密算法。这两种不同类型算法之间的区别主要反映在安全性的定义中：对于公钥加密算法，当攻击者知道加密密钥时也是安全的，而私钥加密算法中却不要求这一点。因此，公钥加密算法允许每个用户公开他们的加密密钥，从而任何其他用户都可以发给他一个加密消息（不需要事先与接收者统一一个私人加密密钥）。下面我们介绍私钥加密算法中的安全性定义。如果将得到加密密钥的攻击者看作附加输入，就可以很容易地得到公钥模拟算法。（对于私钥加密算法，不失一般性，我们可以假设加密密钥与解密密钥是相同的。）

B.1.1 定义

为了简单起见，我们只考虑对单一消息的加密；但这个消息可以比密钥长（这就排除了信息论秘密的情形[200]）。我们介绍安全性的两个等价定义。第一种定义称为语义安全，是 Shannon 的完全安全[200]定义的计算模拟。第二种定义认为，如果分辨任意对的已知消息（例如，全 0 的消息和全 1 的消息）的加密都是不可行的，我们就说这种加密算法是安全的。后一种定义在本质上是技术性的，并被称为加密的不可分辨性。

我们强调这里的定义不仅仅是说由密文恢复出明文是不可行的。这确实是安全加密算法的一个很低的要求，同时我们也声明，这是一个非常弱的要求：加密算法主要用于这样的应用程序，在这种程序中，如果从明文获得特定的部分信息，就会危及整个程序的安全。在设计与程序独立的加密算法时，我们不知道哪一部分信息会危及程序的安全，哪一部分不会。进一步，即使有人想设计加密算法只适合自己的应用程序，（退一步说）他也几乎不可能找到危及这些应用程序的所有可能的部分信息的精确描述。因此，我们要求从密文获得明文的任何信息都是不可行的。再者，在大多数应用程序中明文都不是均匀分布的，且攻击者根据推理可能知道关于它的一些信息。我们要求在这样的情况下，

所有部分信息仍然是保密的。也就是说，即使看到明文的推理信息，要想从密文得到明文的任何新信息都是不可行的(除去能够由推理得到的有关明文的信息)。语义安全的定义假设所有这些都成立。在说明候选构造法的安全性，以及讨论它们作为大协议的组成部分的作用时，加密的不可分辨性的等价定义是非常有用的。

现有的定义 在两种定义中，我们考虑攻击者不仅得到密文，还得到依赖于可能的明文(但不是密钥)的辅助信息。当随机选择好密钥后，用 $E(x)$ 来表示对 x 的加密分布。为了简化说明，我们假设对于安全参数 n，密钥的生成算法产生一个长为 n 的密钥，而加密算法用来对长为 n^2 的消息加密。

定义 B.1.1(语义安全(由[123]得来))：如果对每一个可行算法 A，存在一个可行算法 B，使得对每两个函数 $f,h:\{0,1\}^*\rightarrow\{0,1\}^*$ 和所有二元组 $(X_n,z_n)_{n\in N}$ 的序列，都有

$$\Pr[A(E(X_n)h(X_n),z_n)=f(X_n)]<\Pr[(B(h(X_n),z_n)=f(X_n)]+\mu(n)$$

则称加密算法是语义安全的。

其中 X_n 是 $\{0,1\}^{n^2}$ 上的随机变量，$|z_n|$ 是可行的长度(n 的多项式)，μ 是一个可忽略的函数。而且，B 的复杂性应该与 A 的复杂性相关。

定义 B.1.1 指出，可行的攻击者从密文得不到任何信息。也就是说，当给定一个推理信息(函数 h 给出的)后，它想要计算的任何关于密文的信息(函数 f 给出的)在本质上都与由仅有的推理信息来计算是一样的。特别地，在计算明文的最低有效位或有关明文的任何其他信息时，密文不能提供任何帮助信息。不管明文服从什么分布(由随机变量 X_n 决定)，这一点都是成立的。下面我们看一个等价的定义。

定义 B.1.2(加密的不可分辨性(由[123]得来))：如果对每一个可行算法 A 和所有的三元组 $(x_n,y_n,z_n)_{n\in N}$ 的序列，这里 $|x_n|=|y_n|=n^2$，$|z_n|$ 是可行的长度(n 的多项式)，都有

$$|\Pr[A(E(x_n),z_n)=1]-\Pr[A(E(y_n),z_n)=1]|<\mu(n)$$

其中 μ 是一个可忽略的函数，则称加密算法是加密不可分辨的。

特别地，z_n 可能等于 (x_n,y_n)。这样，要想分辨任意两个像全 0 消息和全 1 消息这样的固定消息的加密都是不可行的。

定理 B.1.3：一个加密算法是语义安全的当且仅当它有不可分辨的加密。

不确定加密 显然，一个安全的公钥加密算法必须使用不确定的(也即随机的)加密算法。否则，给定加密密钥作为一个(附加)输入，就很容易分辨对全 0 消息的加密和对全 1 消息的加密。当考虑加密几条消息(而不是只对一条消息)的安全性时，对私钥加密算法有相同的结论。[1] 这就解释了前述的健壮安全性定义和随机化范例(后面介绍)之间的关联。

B.1.2 构造

基于单向函数的存在性可以构造私钥加密算法。相比之下，熟知的公钥加密算法的构造要求更强的假设(例如陷门置换的存在)。

B.1.2.1 私钥算法

用"伪随机发生器"作为私钥流密码的基础是非常普遍的。我们必须指出，当"伪随机发生

---

[1] 例如，这里使用的确定性加密算法使攻击者能够分辨对两个相同消息的加密和对两个不同消息的加密。

器”很容易被预测时(例如线性同余发生器或它的一些修正算法,它们输出每一个结果数字的位的固定部分[38,84]),这种做法是很危险的。然而只要应用3.3节中定义的伪随机发生器,这种做法就是非常合理安全的。这样,我们得到一个私钥流密码,使得我们能够对明文位的一个流进行加密。注意,这样的一个流密码与我们对加密算法的公式化表示不一致,因为对几条消息加密时,我们必须有一个计数器。换句话说,我们得到一个可变状态的加密算法,对每一条消息加密完毕它都会得到修改。要想得到前面定义的无状态的加密算法,我们可以利用伪随机函数。

基于伪随机函数的私钥加密算法　生成密钥的算法包括选择一个种子 s,这样的函数记为 f_s。为了加密一个消息 $x\in\{0,1\}^n$(用密钥 s),加密算法均匀地选择一个串 $r\in\{0,1\}^n$,得到密文$(r,x\oplus f_s(r))$。为了解密密文(r,y)(用密钥 s),解密算法只需计算 $y\oplus f_s(r)$。证明这种加密算法的安全性分两步进行(在3.6节中建议作为一般的方法论介绍)。

1. 证明算法的一种理想形式是安全的,在此形式中,我们利用均匀选择的函数 $f:\{0,1\}^n\rightarrow\{0,1\}^n$,而不是伪随机函数 f_s。

2. 得出结论,(前面介绍的)真正的随机算法是安全的(否则,我们就可以分辨伪随机函数与真随机函数了)。

注意,如果我们允许加密算法是依赖于历史的(例如,用一个计数器代替 r 的职能),实际上就已经破坏了它的随机性。再者,如果加密算法是用于两个党派之间先入先出通信的,且他们都能够保存计数器的值,则消息发送者就没必要传递计数器的值。

B.1.2.2　公钥算法

这里我们用一个陷门单向置换集合$\{p_\alpha\}_\alpha$和它的一个核心断言 b。

随机化的范例[123]　为了说明这个范例,首先构造一个简单的公钥加密算法。

密钥产生:生成密钥的算法包括随机选择一个置换 p_α 和它的一个陷门;这个置换(或它的描述)作为公钥,而将陷门作为私钥。

加密:要对一个位 σ 加密(用公钥 p_α),加密算法在 p_α 的定义域内均匀地选择一个元素 r,并产生密文$(p_\alpha(r),\sigma\oplus b(r))$。

解密:用私钥对(y,τ)解密,解密算法简单计算 $\tau\oplus b(p_\alpha^{-1}(y))$,其中逆元由陷门(即私钥)来计算。

这个算法对带宽的浪费极大。然而,在构造中潜在的范例在实践中却是非常有价值的。例如,在用RSA加密消息之前随机填充消息(用的填料长度等于消息长度)当然比直接用RSA加密消息要好。如果下面的猜测是正确的,则这样的直接推断就可以应用于商业领域:假设自变量的前 $n/2$ 个最低有效位构成了有 n 位长的模数的RSA的核心函数。那么通过 $n/2$ 个随机位填充消息来对 $n/2$ 位的消息加密,并应用RSA(模数是 n 位)作用于这个结果,这就形成一个安全的公钥加密算法,今后我们称为随机化RSA。

在[35]中将介绍一个候选的公钥加密算法,这个加密算法如下扩充了构造3.4.4(基于单向置换的伪随机发生器的构造)。

密钥产生:同前面一样,生成密钥的算法包括随机选择一个置换 p_α 和它的一个陷门。

加密:(用公钥 p_α)对 n 位长的串 x 加密,加密算法在 p_α 的定义域内均匀地选择元素 s,并产生$(p_\alpha^n(s),x\oplus G_\alpha(s))$密文,其中

$$G_\alpha(s) = b(s) \cdot b(p_\alpha(s)) \cdots b(p_\alpha^{n-1}(s))$$

(我们用记号 $p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x))$，$p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x))$。)

解密:为了用私钥对(y,z)解密,解密算法首先恢复 $s=p_\alpha^{-n}(y)$,然后输出 $z\oplus G_\alpha(s)$。

假设分解布卢姆整数(即,两个与 3 (mod 4)同余的素数的乘积)是很困难的,就可以利用平方求模函数来代替陷门置换,而用最低有效位(记为 lsb)代替它的核心断言[35,5,208,82]。这样就产生了一个安全的公钥加密算法(在图 B.1 中描述),其效率可以与 RSA 的效率相比。回想前面 RSA 算法是不安全的(因为它使用确定的加密算法),而在像分解因式的难处理(或对 RSA 函数的难求逆)这样的标准假设下,随机化的 RSA(前面定义的)还不知道是否是安全的。[①]

> 私钥:两个 $n/2$ 位长的素数 p 和 q,每一个都与 3(mod 4)同余。
>
> 公钥:它们的乘积 $N \stackrel{\text{def}}{=} pq$。
>
> 对消息 $x \in \{0,1^n\}$ 加密:
>
> 均匀选择 $s_0 \in \{1,\ldots,N\}$。
>
> 对 $i=1,\ldots,n+1$,计算 $s_i \leftarrow s_{i-1}^2 \bmod N$ 和 $\sigma_i = \text{lsb}(s_i)$,
>
> 密文是(s_{n+1}, y),其中 $y = x \oplus \sigma_1\sigma_2\cdots\sigma_n$。
>
> 对密文(r,y)解密:
>
> 前面计算的:$d_p = ((p+1)/4)^n \bmod p-1$,$d_p = ((q+1)/4)^n \bmod q-1$,$c_p = q \cdot (q^{-1} \bmod p)$,$c_q = p \cdot (p^{-1} \bmod q)$。
>
> 令 $s' \leftarrow r^{d_p} \bmod p$,$s'' \leftarrow r^{d_q} \bmod q$。
>
> 令 $s_1 \leftarrow c_p \cdot s' + c_q \cdot s'' \bmod N$。
>
> 对 $i=1,\ldots,n$,计算 $\sigma_i = \text{lsb}(s_i)$。和 $s_{i+1} \leftarrow s_i^2 \bmod N$。
>
> 明文是 $y \oplus \sigma_1\sigma_2\cdots\sigma_n$。

图 B.1 布卢姆——戈德华特公钥加密算法[35]

B.1.3 防窃听安全性

前述定义只用于"被动"攻击,即攻击者只是在(传输密文的)通信信道中窃听消息。更强一些的攻击,最厉害的就是所谓的选择密文攻击,在不同的应用程序中都可能发生。而且,这些定义中的攻击都是想提取明文的明确信息的。另一种意图不太明确的定义,即所谓的"可塑性"概念,阐述的是生成一个对相关明文的加密(可能不知道任何原始明文)。这样,我们得到一个攻击"矩阵",其一个维数(参数)是指攻击类型,另一个维数是指攻击目的。

攻击类型 下面对攻击的小型分类显然不是很彻底的.

1. 被动攻击,前面定义中提到的。在公钥算法中,我们又分了两个子类。

(a) 遗忘公钥的被动攻击(前面定义的)。这里用"密钥遗忘性"来指明文的选择不依赖公钥。

---

① 回想前面,如果假设 $n/2$ 个最低有效位构成模数为 n 的 RSA 的核心函数,那么随机化的 RSA 就是安全的。而我们只知道 $O(\log n)$个最低有效位构成模数为 n 的 RSA 的核心函数。

(b) 依赖密钥的被动攻击,在这种攻击中,明文的选择可能会依赖公钥。

(在定义 B.1.1 中明文的选择是指随机变量 X_n,而在定义 B.1.2 中是指串序对(x_n, y_n)。在这两种定义中,明文的选择都不是自适应的。)

2. 选择明文攻击。这里攻击者可以根据自己的选择获得任意明文的加密密文(在密钥受到攻击的情况下)。在公钥算法下这种攻击起不到任何作用。

3. 选择密文攻击。这里攻击者可以根据自己的选择获得对任意密文的解密(在密钥受到攻击的情况下)。也就是说,攻击者得到与所用的解密密钥相应的解密函数的预言访问。我们将这种攻击分为两类:

(a) 在一个先验选择密文攻击中,攻击者在知道它要攻击的密文(即那些它必须获得部分信息或者是形成相关密文的密文)之前就得到了这个预言访问。也就是说,攻击包括两个阶段:在第一个阶段,攻击者得到预言访问,第二个阶段预言被移走,攻击者得到一个"测试密文"(即,一个与不可塑性相矛盾的需掌握或修正的目标)。

(b) 在一个后验选择密文攻击中,攻击者首先得到一个要攻击的目标密文,但是它对预言的访问因为它不能做等同于目标密文的质疑而受到限制。

在这两种情况下,攻击者可以做与合法密文不一致的质疑,并得到相应的回答(即一个特殊的"失败"符号)。

攻击目的 与前面一样,下面的结论也是不彻底的。

1. 标准安全:获取关于明文的信息的不可行性。和前面定义的一样,这样的信息一定是作用于赤裸的明文上的一个函数(或随机过程),且是不依赖于加密(或解密)密钥的。

2. 相比之下,不可塑性的概念[64]是指生成既依赖于明文又依赖于当前加密密钥的串。特别地,它要求当给定一个密文时,攻击者为相关明文生成有效密文是不可行的。例如,给定一个与形式为 $1x$ 的明文相对应的密文,产生形式为 $0x$ 的明文的一个密文是不可行的。

除了私钥算法中的被动攻击,不可塑性就意味着在抵制获取明文信息时是安全的。在后验选择密文攻击中,安全性与不可塑性是等价的(比较[64,16])。要想得到安全的私钥加密和公钥加密算法中各种概念之间的关系的详细讨论,读者可分别参考[142]和[16]。

一些已知的构造 与基本情况中一样,(强安全的)私钥加密算法可以基于单向函数的存在性进行构造,而(强安全的)公钥加密算法可以基于陷门置换的存在性进行构造。

私钥算法 对于先验选择密文攻击,基于伪随机函数构造的私钥加密算法(如前所述)是安全的。[①]

通过采用基本加密算法之后的消息认证算法[②],可以很容易地将任何被动的安全私钥加密算法转化成在后验选择密文攻击下安全的算法。

公钥算法 对先验选择密文攻击算法安全的公钥加密算法,可以通过假设存在陷门置换并利用非交互零知识证明[176]来构造(回想前面,后面的证明系统可以在前一假设的基础上构造)。

对后验选择密文攻击算法安全的公钥加密算法,也可以在同样的假设下构造[64],但是这

---

① 在后验选择密文攻击中这个算法是不安全的:当输入一个密文$(r, x\oplus f_s(r))$时,我们通过质疑(r, y')得到 $f_s(r)$,其中 $y' \neq x\oplus f_s(r)$。这个质疑用满足 $y' = x'\oplus f_s(r)$的 x' 来回答。

② 见 B.2 节中的定义。

种构造更为复杂。

事实上，两种对选择密文攻击安全的公钥加密算法的构造都被认为是合理的结果（还可以提供一些有用的构造范例）。在广泛接受的假设前提下提供对后验选择密文攻击安全的“有合理效率”的公钥加密算法，是一个很重要的公开未决问题。[①]

B.1.4 一些建议

B.1.4.1 关于进一步阅读的建议

可以在[99]上获得相关章节的部分原始草稿。

此外，原始文献如下：文献[123]中有一个很好的触发性讨论，但我们推荐[92,94]中更明确的处理方法，如果采用非均匀复杂性措施（如前面所做的），就能把这种处理方法充分化简（如上所做）[②]。公钥加密算法的构造（如上面所述）的进一步讨论在[123,92,35,5]中介绍。对不可塑密码学的讨论，详见[64]，实际上是对加密算法定义域的一个扩展。

B.1.4.2 教学建议

我们建议重点要集中在安全的基本概念（在 B.1.1 小节和 B.1.2 小节中介绍）上：介绍两种定义，证明它们是等价的，并讨论为满足这些定义在加密过程中使用随机性的必要性。接着介绍在 B.1.2 小节描述的所有构造。我们相信网上[99]中的草稿提供了所有这些内容的详细细节。

B.2 签名：摘要

同样有私钥和公钥两种情况，它们都包括 3 个有效算法：生成密钥算法、签名和验证算法（私钥签名算法通常指消息认证算法或编码（MAC））。两种类型的不同之处同样反映在安全的定义中。这种不同决定了它们不同的职能（比加密算法中的差别还大）：公钥签名算法（今后指签名算法）用来生成可普遍验证的签名（给定签名者使用的公钥）；私钥签名算法（今后指消息认证算法）专门用来鉴别在一个互相信任的用户集合中传送的消息（因为验证签名的能力与产生签名的能力有关）。换句话说，消息认证算法用来鉴别成员（专指两个）之间传送的信息，目的是让接收者确信消息确实是由合法发送者发送的。特别地，消息认证算法不能让第三者确信消息是由合法发送者发送的（而不是接收者自己产生的）。相比之下，公钥签名可以用来向第三者证实：对某文档的签名是发送给第二个成员的，因此，将来这个成员就可以向第三者证实这个文档确实由签名者产生、发送并验证过。

---

① 文献[57]中的“有合理效率”算法基于关于 Diffie-Hellman 密钥交换的强假设。特别地，假设对于一个素数 P 和本原元 g，给定 $(P,g,(g^x \bmod P),(g^y \bmod P),(g^z \bmod P))$，决定 $z \equiv xy \pmod{P-1}$ 是否成立是不可行的。

② 注释：在提供加密算法的安全性的均匀复杂性处理时，[92]是由[94]得来的。

B.2.1 定义

我们考虑对签名算法的一种很强的攻击和破解它的一个很自由的概念。特别地，攻击者可以根据自己的选择访问任何消息的签名。大家可能会说，在许多应用程序中，这样一般化的攻击是不可能的(因为签名的消息肯定有特定的格式)。然而我们这样认为是因为我们以为定义一般化的容许(即，独立于应用程序)消息是不可能的，从而攻击的一般化/健壮的定义必须和这里建议的一样(我们的方法最坏也只能说是过分小心了)。同理，如果攻击者能够产生任何消息的有效签名，对于这些消息在攻击的时候不要求它们的签名，我们就说攻击是成功的。此外，这里定义产生对可能的"非感性"消息的签名的能力为能否攻破这个算法。然而，我们决不会有"有意义的"消息的一般化(即，独立于应用程序)概念(因此只有伪造签名可以被认为是对算法的一个破坏)。

定义 B.2.1(不可伪造签名[125])：

- 一个**选择明文攻击**是一个过程，当输入一个验证密钥时，它能够根据自己的选择获得对消息的签名(与相应的签名密钥对应)。
- 如果它输出一个消息的有效签名，而对于这个消息，在攻击过程中不要求签名，我们就说这个攻击是**成功**的(在存在伪造的情况下)。
- 如果每个可行的选择明文攻击只能以可忽略的概率成功，则称这个签名算法是**安全**的(或不可伪造的)。

我们强调在前述的定义下，普通 RSA(像 Rabin 算法[187]和 DSS[169]的普通版本)都是不安全的。然而，在应用 RSA(或另一个算法)前，如果消息是"随机"的，则它可能是安全的[22]。因此，随机化范例在这里也是非常关键的。

除去攻击者不能得到验证密钥作为输入之外，消息认证算法的安全性定义与此相似。

B.2.2 构造

消息认证算法和签名算法都可以基于单向函数的存在而构造。

B.2.2.1 消息认证

利用伪随机函数可以构造消息认证算法[103]：为了用密钥 s 验证加密的消息 x，可以生成一个标记 $f_s(x)$，其中 f_s 是与 s 相关的伪随机函数。认证的方式与此类似。然而，正如[15]中说的那样，对获得消息认证来说，伪随机函数的广泛使用似乎是不必要的，且基于其他的密码元可以获得更有效的算法。下面我们介绍两种方法：

1. 只要攻击者不能获悉算法的输出，就可以利用防伪造的安全算法采集消息的指纹(例如，用通用的哈希法[49])，并利用不可塑算法(例如，一个私钥加密算法或一个伪随机函数)"隐藏"结果(在特定的情况下[209]，对不可塑性可不做要求)。

2. 利用无冲突算法来哈希消息[58,59]，并用作用于(短的)有固定长度的串上的 MAC 来验证结果[15]。

B.2.2.2 签名算法

签名算法的构造中,3 个重要的范例是对"有效的"签名密钥的"更新"、"验证树"的使用和"哈希范例"。

更新的范例[125] 为了说明这个范例,假设有一个能很好地防止"随机消息攻击"(即,在这种攻击中,攻击者只获得随机选择消息的签名)的签名算法。进一步假设我们有一次签名算法(即,签名算法能很好地防止这种攻击,在攻击中,攻击者根据自己的选择获得对单一消息的一个签名)。我们可以按如下步骤获得一个安全的签名算法:当需要给一个新消息签名时,我们为一次签名算法生成一个新的随机签名密钥,用它来为消息签名,并用主签名算法(对"随机消息攻击"是健壮的)[①]的固定签名密钥给相应的(一次)验证密钥签名[71]。(这里使用的)一次签名算法是很容易构造的(例如[161])。

验证树范例[160,125] 为了说明这个范例,我们介绍如何仅利用一次签名算法来构造一般化的签名算法(可以用一个 n 位长的验证密钥对 $2n$ 位长的串签名)。思想是用一个初始签名密钥(即,与公共验证密钥相对应的密钥)来签名/鉴别两个新/随机的验证密钥。这两个相应的验证密钥又用来签名/鉴别 4 个新/随机的验证密钥(每一个签两个),一直如此下去。在 l 个这样的步骤之后,这个过程形成一个有 2^l 个叶子的二叉树,一个叶子对应一次签名算法的一个实例。叶子上的签名密钥可以用来对实际消息签名,从而可以用由根节点出发的路径来鉴别相应的验证密钥。也就是说,对一个新消息签名,我们如下进行。

1. 在树上分配一个新的叶子。这要求要么保持如此签名的消息数目的一个计数器,要么随机地选择一个叶子(假设叶子的数量远远大于要签名的消息数目的平方)。

2. 生成或从存储库得到与从根到所选择的叶子路径上的每个节点相对应的签名/认证密钥对,以及这些节点的兄弟对应的密钥对。也就是说,令 $v_0,v_1,\ldots,v_l$ 表示沿着从根节点 v_0 到所选叶子 v_l 的路径上的节点,并令 u_i 是 v_i 的兄弟(对 $i=1,\ldots,l$)。然后我们生成/检索每一个 u_i 和 v_i 的密钥对(对 $i=1,\ldots,l$)。

在对两个不同的消息签名的过程中,当遇到相同的节点时要使用相同的密钥对。

3. 用与选择的叶子 v_l 相对应的签名密钥对消息签名。用与父亲节点相对应的签名密钥对沿着上述路径的每一对与内部结点的孩子节点相对应的验证密钥签名。也就是说,对于每一个 $i=1,\ldots,l$,用与节点 v_{i-1} 相对应的签名密钥对 u_i 和 v_i 的验证密钥签名。

将所有这些签名(和相应的认证密钥)串联起来就得到想要的签名。回想前面,与根节点相对应的密钥对实际上就是签名算法的密钥对;也就是说,验证部分放到了公共文件中,根节点的孩子的验证密钥签名(相对于根节点的签名密钥)是所有签名的一部分。

使用伪随机函数可以避免存储前面签名中使用的节点的值[89]。

应用这个范例,并假设 RSA 函数是难求逆的,可以得到一个安全的签名算法[125,89](使用一个签名消息数目的计数器),在这个算法中,签名/验证第 i 个消息的时间是 $2\log_2 i$,比普通的 RSA 函数要慢。利用有大输入端的一个树(仍然假设 RSA 函数是难求逆的),可以获得

---

① 或者,可以提前生成一次密钥对和对验证密钥的签名,从而引出"下线/在线"签名算法。

一个安全的签名算法[67,56],它对于合理参数,只比普通的 RSA 函数慢 5 倍。[①] 我们强调普通的 RSA 不是一个安全的签名算法,而它的随机化版本(前面提到的)也不知道是否可以还原为 RSA 的难求逆性。

哈希范例 常见实例是通过二阶段过程对一个真实文档签名:首先,文档被分成相对短的位串,然后对此结果串应用基本的签名算法。只要这个哈希函数是无冲突的(在[58]中定义),这个启发式方法就是正确的。假设存在无爪函数集合(见定义 2.4.6),就可以构造无冲突哈希函数[58]。确实可以假设一些经常使用的函数是无冲突的(例如,MD5 或 SHA),但这种假设必须通过检测(确实可能出现检测结果是错误的这种情况)。我们强调,在上述两阶段过程中不仔细估计哈希算法是否为无冲突的就使用它是很危险的。

上述范例中一个很有用的变化是使用通用的单向哈希函数([175]中定义),而不是使用前面的无冲突哈希函数。在这种情况下,对算法的每一个应用程序都选择一个新的哈希函数,在哈希函数的简洁描述和结果哈希串中都要应用基本的签名算法(相比之下,当应用无冲突的哈希函数时,签名算法的所有应用程序都使用一个相同的函数,其中此函数的描述是签名者的公钥)。使用通用的单向哈希函数的优点是,它们的安全性要求比使用无冲突函数时要弱(例如,前者利用任意单向函数都可以构造[192],而这一点后者还不是很明确)。

定理 B.2.2(似乎合理的结果[175,192]):*签名算法存在当且仅当单向函数存在。*

与前面的范例(和其中的一些构造)不同,大家熟知的由任意单向函数构造签名算法的方法是没有任何实践意义的。找一个既有实践意义,又利用任意单向函数的可选构造确实是一个非常重要的公开未决问题。

B.2.3 一些建议

B.2.3.1 进一步阅读的建议

上网在[100]中可以找到签名算法的初稿部分。

此外,还有一些原始文献:要想明确了解签名算法,读者可参考[125]和[183]。一些易懂的构造方法在[20,71,67]中介绍。定理 *B*.2.2 的证明可以从[175,192]中提取出来:第一篇文献介绍了基本的方法并应用任意的单向置换实现了这种方法,第二篇文献指出如何利用任意单向函数实现这种方法。基本模型中的变体在[183]和[50,137]中讨论。对消息认证算法(*MAC*)的讨论,读者请参阅[15]。

B.2.3.2 教学建议

我们建议重点要集中在签名算法上,介绍其主要的定义和一些构造。你可以利用[125]做定义方面的处理,但不要用它作为构造,其内在的根本思想在文献[20]和[175]中体现得很明显。实际上,我们建议介绍[175]中签名算法的一个变体,利用无冲突哈希函数(比较[58])代替通用的单向哈希函数(比较[175])。这允许我们在少数的讲稿中介绍许多重要的范例和技

---

① 这个数字指的是加密 1 000 000 000 个消息。[67]中的算法要求一个通用的系统参数集合,包括 1000～2000 个与模数有相同长度的整数,在[56]的算法中,这个要求又被去掉了。

术(例如,更新范例、鉴别树、哈希范例和一次签名算法)。我们相信在[100]中提供的草稿能为这些问题提供非常详细的介绍。

消息认证的基本处理(即,目的、定义和基于伪随机函数的构造)一次授课就可以完成,也可以参考[100](然而,这并不包括更有效的消息认证算法中所用到的可选构造方法)。

B.3 密码学协议:摘要

考虑密码(协议)问题的基本框架实际上就是指定一个由 n 输入映射到 n 输出的随机过程。这个过程的输入被认为是 n 个用户的局部输入,n 个输出是它们相应的局部输出。随机过程描述想得到的性能。也就是说,如果 n 个用户互相信任(或信任某一外面的用户),则每一个都可以发送消息给信任的用户,这个用户可以计算过程的结果,并发送相应的输出给每一个用户。本节的问题是现实中互不信任的用户到底能信任到何种程度。

B.3.1 定义

为了简单起见,我们考虑指定的过程是确定的过程,且 n 个输出相同的特殊情况。也就是说,我们考虑一个任意的 n 元函数,n 个用户希望获得与它们的 n 个输入相对应的函数值。每一个用户都希望自己获得正确的函数值,并能阻止其他用户获得任何其他信息(即,除函数值和它隐含的信息之外的任何信息)。

我们首先观察(不可避免的一件事)到,每一个用户在执行协议之前都可以交换它们的局部输入。即使当所有用户都共用一个可信任的中介时,这仍然是不可避免的。总的来说,在安全的多用户计算①的定义中的基本范例实际上是指在现实协议中可能出现,在理想模型中(所有用户可以共有一个可信任的中介)可以模拟的情况。因此,安全协议中用户之间的"有效屏障"是限定在相应的理想模型的假设之下的。在真实计算中,对用户的特定限制和要求不同,就会有不同的定义。这典型地反映在相应的理想模型的定义中;详情请参见下面的例子。

B.3.1.1 一个例子:在大多数人诚实的假设下的计算

下面我们考虑一个理想模型,在这个模型中,任意少数的用户都可能进行如下的密谋。首先,这少数几个人共享其原始输入并共同决定一个替代输入②给可信任的中介(而其他用户将他们各自的输入发送给大家都信任的中介)。当中介返回他们的输出时,大多数用户中的每个人都分别地输出结果,而那些密谋的少数用户就可以利用他们知道的信息(即,输出和他们中所有用户的原始输入)来计算一个输出。一个安全的大多数人诚实的多用户计算必须能模拟

① 目前我们对定义问题的理解绝大多数来自[165]未完成草稿中的高水平讨论。在[11,12]中介绍了一个类似的定义方法。[122]中的方法更为一般:它避免了安全(官能度)的定义,但给出了协议健壮性概念的一个相关定义。[165,11,12]中的定义方法的一个小例在[45]中介绍,并被证明是与主要概念一致的。

② 如果用户的局部输入被其他用户验证,这个替代就可以避免。在这种情况下,(理想模型中)的用户就有两种选择,既可以用它正确的局部输入来执行协议也可以根本不执行协议(但不能用一个替代输入来执行协议)。模拟这种理想模型的安全协议也可以构造。

这个理想模型。也就是说，在现实协议中任何可行的控制少数用户的攻击效果在本质上都可以由理想模型中(不同的)控制相应的少数用户的攻击来模拟。这就意味着，在安全的协议中每个少数群体的影响"在本质上都局限于"在协议开始前替换它们的局部输入(与大多数其他的局部输入相独立)，并在协议终止后替换它们的局部输出(仅依赖于它们自己的局部输入和输出)。(我们强调，在真实的执行过程中少数用户确实可以获得额外的信息；但在一个安全的协议中，它们实际上从这些信息中得不到任何东西。)

依据此定义的安全协议，甚至可以抵御少数用户异常终止协议。(在现实协议中)一个异常终止协议的用户(在理想模型中)可以由这样的一个用户来模拟，此用户既可以在向可信任的中介提供它的输入之前终止协议(此情况下用的是一个错误的输入)，也可以在提供输入之后终止。在两种情况下，(在现实协议中)即使少数用户终止了协议，多数用户仍可以计算输出。当然，没有大多数用户的诚实保证(例如，在双用户计算中)，这种情况是不可能出现的[53]。

B.3.1.2　另一个例子：允许异常终止的双用户计算

如前所述，我们考虑一个理想模型，模型中两个用户中的每一个都能够在任何点及时"中断"与可信任的第三方(中介)的联系。特别地，这种情况发生在可信任的中介将通信输出发送给一个用户之后，却还没有发送给另一个用户之前。一个安全的允许异常终止的双用户计算必须能够模拟这个理想模型。也就是说，在这个安全协议中，每个用户的"有效故障"限制在根据自己的选择提供最初的输入，并及时在任意点终止计算。我们强调，同前面一样，每个用户最初的输入选择与其他用户的选择是独立的。

将前面的内容一般化，我们可以考虑安全的允许异常终止的多用户计算。这里，在理想模型中，每一个用户都可以在任意点"异常终止"与可信任的第三方的联系；特别地，这种情况可能发生在第三方对一些用户提供了计算结果，但还没有对所有的用户提供结果之后。

B.3.2　构造

定理 B.3.1(一般可能结果的不严格叙述)：*假设存在陷门置换。那么*

- *任意多用户官能度都可以在一个允许异常终止的模型中被安全地计算*(比较[211]中的双用户情况和[113]中的多用户情况)。
- *只要严格多数的用户是诚实的，任何多用户官能度都可以被安全地计算*[112,113]。

每一条的证明都可以分两步进行[98]。

1. 介绍"半诚实"模型下的安全协议，在这种模型中，坏用户也遵守协议，只不过他们同时还记录中间结果。①

一个关键的思想是，考虑沿着一个有线电路(计算想要的函数)从输入线到输出线的值的传送。每个用户与其他所有用户通过一个秘密的共享算法来共享自己的输入，就意味着协议的开始，因此这些共享输入的任何真子集都不会有共享秘密的任何信息(例如，每个用户都得

① 换句话说，当只给诚实用户局部输入和输出时，必须模拟不诚实用户的局部思想。实际上，这个模型是与零知识的诚实验证模型相对应的(见 4.3.1.7 节)。

到一个均匀选择的值，则庄家的那一份就等于其他所有用户值的异或）。一个典型的步骤就是从某个门的输入线的值安全计算此门的输出线的值。也就是说，m 个用户为计算一个随机的 m-用户官能度 $((a_1,b_1),\ldots,(a_m,b_m))\rightarrow(c_1,\ldots,c_m)$ 使用一个安全协议，其中每一个 c_i 服从均匀分布，且满足 $\bigoplus_{i=1}^{m}c_i=\text{gate}(\bigoplus_{i=1}^{m}a_i,\bigoplus_{i=1}^{m}b_i)$。对这个电路中的每一个门（以合适的顺序）重复上述步骤，沿着电路线从电路的输入端到输出端，用户安全地传送他们的值。在传送过程的终端，每个用户宣布其在电路输出线上的值，从而产生实际输出。因此，安全计算一个任意的官能度（可能是很复杂的）就化简为安全计算几个特定的简单的官能度（即，给定一个布尔门的输入值，安全计算此门的输出的随机值）。事实上，计算这些简单官能度的安全协议也已经给出。

2. 将半诚实模型中的安全协议转化为成熟的安全协议。这里，为了"迫使"每一个用户都正确地执行协议（即，与在半诚实模型中一样），我们使用零知识证明和公平掷币协议。

公平掷币协议是用一个不可遗忘的承诺算法构造的（见 4.9.2 小节），而后者反过来又用到零知识证明的知识（见 4.7 节）。

假设陷门置换存在，我们重点考虑这些构造的一般性质，并将它们看作是似乎合理的结果，即断言许多密码学问题是可解决的。正如零知识证明中所讨论的，这些一般结论的价值是，使得一个人能很容易地推断出他所面临的问题在原则上是可解决的（特别是很容易在这个框架中解决问题）。但是，我们不提倡用由这些一般结论得出的解决方法；宁可抓住问题的细节，并用这些一般结论中的观点或技巧来解决。[①]

在不同的模型中可得到类似的可能结论。特别地，我们介绍在私人信道模型[27,51]和易变攻击[182]中的安全计算。

B.3.3 一些建议

B.3.3.1 关于进一步阅读的建议

关于这个测量资料的初稿在[98]中可以找到。草稿提供了有关基本定义和结论的说明，以及结论的详细证明。对一些定义方面的问题的限制性讨论见[11,12,44,45,122,165]；我们的建议是先读[45]。

B.3.3.2 教学建议

这个领域非常复杂，因此我们建议只对一些定义和构造略做介绍就行了。特别地，我们建议拿两种背景（即，多数诚实或双用户计算）中的一种来略述定义和构造。我们的选择是双用户下的情况；尽管（允许异常终止）的定义是非常复杂的（但基于相对少的概念下的简单记号和简单构造足以弥补这一点）。我们建议：重点要强调定义的方法（即通过理想模型攻击中"与一个可信任的中介竞争"来模拟现实模型中的攻击），还要重点介绍构造中的主要思想（可以略去少数一些）。我们相信[98]中的草稿会为这些内容提供足够详细的介绍。

① 例如，尽管门限密码学（比较[62,87]）只是多用户计算的一个特例，但掌握它的细节仍然是非常有用的。

参考文献

[1] L.M. Adleman and M. Huang. *Primality Testing and Abelian Varieties Over Finite Fields*. Springer-Verlag Lecture Notes in Computer Science (Vol. 1512), 1992. (Preliminary version in *19th ACM Symposium on the Theory of Computing*, 1987.)

[2] W. Aiello and J. Håstad. Perfect Zero-Knowledge Languages Can Be Recognized in Two Rounds. In *28th IEEE Symposium on Foundations of Computer Science*, pages 439–448, 1987.

[3] M. Ajtai. Generating Hard Instances of Lattice Problems. In *28th ACM Symposium on the Theory of Computing*, pages 99–108, 1996.

[4] M. Ajtai, J. Komlos, and E. Szemerédi. Deterministic Simulation in LogSpace. In *19th ACM Symposium on the Theory of Computing*, pages 132–140, 1987.

[5] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA/Rabin Functions. Certain Parts Are as Hard as the Whole. *SIAM Journal on Computing*, Vol. 17, April, pages 194–209, 1988.

[6] N. Alon and J.H. Spencer. *The Probabilistic Method*. Wiley, 1992.

[7] T.M. Apostol. *Introduction to Analytic Number Theory*. Springer, 1976.

[8] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–420, 1985.

[9] E. Bach. *Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms*. ACM Distinguished Dissertation (1984), MIT Press, Cambridge, MA, 1985.

[10] E. Bach and J. Shallit. *Algorithmic Number Theory, Vol. I: Efficient Algorithms*. MIT Press, Cambridge, MA, 1996.

[11] D. Beaver. Foundations of Secure Interactive Computing. In *Crypto91*, Springer-Verlag Lecture Notes in Computer Science (Vol. 576), pages 377–391, 1992.

[12] D. Beaver. Secure Multi-Party Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. *Journal of Cryptology*, Vol. 4, pages 75–122, 1991.

[13] M. Bellare. A Note on Negligible Functions. Tech. Rep. CS97-529, Department of Computer Science and Engineering, UCSD, March 1997.

[14] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security. In *37th IEEE Symposium on Foundations of Computer Science*, pages 514–523, 1996.

[15] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Crypto96*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1109), pages 1–15, 1996.

[16] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Crypto98*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1462), pages 26–45, 1998.

[17] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 390–420, 1992.

[18] M. Bellare, S. Halevi, A. Sahai, and S. Vadhan. Trapdoor Functions and Public-Key Cryptosystems. In *Crypto98*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1462), pages 283–298, 1998.

[19] M. Bellare, R. Impagliazzo, and M. Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.

[20] M. Bellare and S. Micali. How to Sign Given Any Trapdoor Function. *Journal of the ACM*, Vol. 39, pages 214–233, 1992.

[21] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *1st Conference on Computer and Communications Security*, ACM, pages 62–73, 1993.

[22] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EuroCrypt96*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1070), pp. 399–416, 1996.

[23] M. Bellare and M. Yung. Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation. *Journal of Cryptology*, Vol. 9, pages 149–166, 1996.

[24] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the Theory of Average Case Complexity. *Journal of Computer and System Science*, Vol. 44, No. 2, April, pages 193–219, 1992.

[25] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything Provable Is Probable in Zero-Knowledge. In *Crypto88*, Springer-Verlag Lecture Notes in Computer Science (Vol. 403), pages 37–56, 1990.

[26] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th ACM Symposium on the Theory of Computing*, pages 113–131, 1988.

[27] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[28] E.R. Berlekamp. Factoring Polynomials over Large Finite Fields. *Mathematics of Computation*, Vol. 24, pages 713–735, 1970.

[29] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory*, 1978.

[30] M. Blum. How to Exchange Secret Keys. *ACM Trans. Comput. Sys.*, Vol. 1, pages 175–193, 1983.

[31] M. Blum. Coin Flipping by Phone. In *24th IEEE Computer Conference (CompCon)*, February, pages 133–137, 1982. (See also *SIGACT News*, Vol. 15, No. 1, 1983.)

[32] L. Blum, M. Blum, and M. Shub. A Simple Secure Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, Vol. 15, pages 364–383, 1986.

[33] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, pages 1084–1118, 1991. (Considered the journal version of [34].)

[34] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988. (See [33].)

[35] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which Hides All Partial Information. In *Crypto84*, Springer-Verlag Lecture Notes in Computer Science (Vol. 196), pages 289–302, 1985.

[36] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. (Preliminary version in *23rd IEEE Symposium on Foundations of Computer Science*, 1982.)

[37] R. Boppana, J. Håstad, and S. Zachos. Does Co-NP Have Short Interactive Proofs? *Information Processing Letters*, Vol. 25, May, pages 127–132, 1987.

[38] J.B. Boyar. Inferring Sequences Produced by Pseudo-Random Number Generators. *Journal of the ACM*, Vol. 36, pages 129–141, 1989.

[39] G. Brassard. A Note on the Complexity of Cryptography. *IEEE Transactions on Information Theory*, Vol. 25, pages 232–233, 1979.

[40] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37, No. 2, pages 156–189, 1988. (Preliminary version by Brassard and Crépeau in *27th IEEE Symposium on Foundations of Computer Science*, 1986.)

[41] G. Brassard and C. Crépeau. Zero-Knowledge Simulation of Boolean Circuits. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 223–233, 1987.

[42] G. Brassard, C. Crépeau, and M. Yung. Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols. *Theoretical Computer Science*, Vol. 84, pages 23–52, 1991.

[43] E.F. Brickell and A.M. Odlyzko. Cryptanalysis: A Survey of Recent Results. In *Proceedings of the IEEE*, Vol. 76, pages 578–593, 1988.

[44] R. Canetti. *Studies in Secure Multi-Party Computation and Applications*. Ph.D. thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, June 1995. (Available from `http://theory.lcs.mit.edu/~tcryptol/BOOKS/ran-phd.html`.)

[45] R. Canetti. Security and Composition of Multi-party Cryptographic Protocols. *Journal of Cryptology*, Vol. 13, No. 1, pages 143–202, 2000.

[46] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. In *30th ACM Symposium on the Theory of Computing*, pages 209–218, 1998.

[47] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. In *32nd ACM Symposium on the Theory of Computing*, pages 235–244, 2000.

[48] E.R. Canfield, P. Erdos, and C. Pomerance. On a Problem of Oppenheim Concerning "factorisatio numerorum." *Journal of Number Theory*, Vol. 17, pages 1–28, 1983.

[49] L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Science*, Vol. 18, pages 143–154, 1979.

[50] D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto82*, pages 199–203, Plenum Press, New York, 1983.

[51] D. Chaum, C. Crépeau, and I. Damgård. Multi-party Unconditionally Secure Protocols. In *20th ACM Symposium on the Theory of Computing*, pages 11–19, 1988.

[52] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *26th IEEE Symposium on Foundations of Computer Science*, pages 383–395, 1985.

[53] R. Cleve. Limits on the Security of Coin Flips When Half the Processors Are Faulty. In *18th ACM Symposium on the Theory of Computing*, pages 364–369, 1986.

[54] J.D. Cohen and M.J. Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme. In *26th IEEE Symposium on Foundations of Computer Science*, pages 372–382, 1985.

[55] A. Cohen and A. Wigderson. Dispensers, Deterministic Amplification, and Weak Random Sources. In *30th IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[56] R. Cramer and I. Damgård. New Generation of Secure and Practical RSA-based Signatures. In *Crypto96*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1109), pages 173–185, 1996.

[57] R. Cramer and V. Shoup. A Practical Public-Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attacks. In *Crypto98*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1462), pages 13–25, 1998.

[58] I. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In *EuroCrypt87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 304), pages 203–216, 1988.

[59] I. Damgård. A Design Principle for Hash Functions. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 416–427, 1990.

[60] I. Damgård. Concurrent Zero-Knowledge Is Easy in Practice. Theory of Cryptography Library, 99-14, June 1999. `http://philby.ucsd.edu/cryptolib`.

[61] I. Damgård, O. Goldreich, T. Okamoto, and A. Wigderson. Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs. In *Crypto95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 963), pages 325–338, 1995.

[62] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 307–315, 1990.

[63] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22 (Nov.), pages 644–654, 1976.

[64] D. Dolev, C. Dwork, and M. Naor. Non-malleable Cryptography. In *23rd ACM Symposium on the Theory of Computing*, pages 542–552, 1991. (Full version available from authors.)

[65] D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, Vol. 30, No. 2, pages 198–208, 1983.

[66] C. Dwork, U. Feige, J. Kilian, M. Naor, and S. Safra. Low Communication Perfect Zero Knowledge Two Provers Proof Systems. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 215–227, 1992.

[67] C. Dwork and M. Naor. An Efficient Existentially Unforgeable Signature Scheme and its Application. *Journal of Cryptology*, Vol. 11, No. 3, pages 187–208, 1998.

[68] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. In *30th STOC*, pages 409–418, 1998.

[69] S. Even and O. Goldreich. On the Security of Multi-party Ping-Pong Protocols. In *24th IEEE Symposium on Foundations of Computer Science*, pages 34–39, 1983.

[70] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *CACM*, Vol. 28, No. 6, pages 637–647, 1985.

[71] S. Even, O. Goldreich, and S. Micali. On-line/Off-line Digital Signatures. *Journal of Cryptology*, Vol. 9, pages 35–67, 1996.

[72] S. Even, A.L. Selman, and Y. Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Information and Control*, Vol. 61, pages 159–173, 1984.

[73] S. Even and Y. Yacobi. Cryptography and NP-Completeness. In *Proceedings of 7th ICALP*, Springer-Verlag Lecture Notes in Computer Science (Vol. 85), pages 195–207, 1980. (See [72].)

[74] U. Feige. Error Reduction by Parallel Repetition – The State of the Art. Technical Report CS95-32, Computer Science Department, Weizmann Institute of Science, Rehovot, Israel, 1995.

[75] U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, Vol. 1, pages 77–94, 1988.

[76] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs under General Assumptions. *SIAM Journal on Computing*, Vol. 29, No. 1, pages 1–28, 1999.

[77] U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 526–544, 1990.

[78] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd ACM Symposium on the Theory of Computing*, pages 416–426, 1990.

[79] W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, New York, 1968.

[80] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 186–189, 1987.

[81] M. Fischer, S. Micali, C. Rackoff, and D.K. Wittenberg. An Oblivious Transfer Protocol Equivalent to Factoring. Unpublished manuscript, 1986. (Preliminary versions were presented in *EuroCrypt84* and in the *NSF Workshop on Mathematical Theory of Security*, Endicott House (1985).)

[82] R. Fischlin and C.P. Schnorr. Stronger Security Proofs for RSA and Rabin Bits. In *EuroCrypt97*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1233), pages 267–279, 1997.

[83] L. Fortnow. The Complexity of Perfect Zero-Knowledge. In *19th ACM Symposium on the Theory of Computing*, pages 204–209, 1987.

[84] A.M. Frieze, J. Håstad, R. Kannan, J.C. Lagarias, and A. Shamir. Reconstructing Truncated Integer Variables Satisfying Linear Congruences. *SIAM Journal on Computing*, Vol. 17, pages 262–280, 1988.

[85] O. Gaber and Z. Galil. Explicit Constructions of Linear Size Superconcentrators. *Journal of Computer and System Science*, Vol. 22, pages 407–420, 1981.

[86] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[87] P.S. Gemmell. An Introduction to Threshold Cryptography. In *CryptoBytes* (RSA Laboratories), Vol. 2, No. 3, 1997.

[88] R. Gennaro and L. Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic Constructions. *ECCC*, TR00-022, May 2000.

[89] O. Goldreich. Two Remarks Concerning the GMR Signature Scheme. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 104–110, 1987.

[90] O. Goldreich. Towards a Theory of Software Protection and Simulation by Oblivious RAMs. In *19th ACM Symposium on the Theory of Computing*, pages 182–194, 1987.

[91] O. Goldreich. *Foundation of Cryptography – Class Notes*. Preprint, spring 1989. (Superseded by the current book in conjunction with [92].)

[92] O. Goldreich. *Lecture Notes on Encryption, Signatures and Cryptographic Protocol*. (Extracts from [91]. Available from `http://theory.lcs.mit.edu/~oded/ln89.html`. Superseded by the combination of [99], [100], and [98].)

[93] O. Goldreich. A Note on Computational Indistinguishability. *Information Processing Letters*, Vol. 34, May, pages 277–281, 1990.

[94] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, Vol. 6, No. 1, pages 21–53, 1993.

[95] O. Goldreich. *Foundation of Cryptography – Fragments of a Book*. February 1995. (Available from `http://theory.lcs.mit.edu/~oded/frag.html`. Superseded by the current book in conjunction with [99].)

[96] O. Goldreich. Notes on Levin's Theory of Average-Case Complexity. *ECCC*, TR97-058, December 1997.

[97] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Algorithms and Combinatorics Series (Vol. 17), Sprirger-Verlag, 1999.

[98] O. Goldreich. *Secure Multi-Party Computation*. (In preparation, 1998. Working draft available from `http://theory.lcs.mit.edu/~oded/gmw.html`.)

[99] O. Goldreich. *Encryption Schemes – Fragments of a Chapter*. (December 1999. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-book.html`.)

[100] O. Goldreich. *Signature Schemes – Fragments of a Chapter*. (May 2000. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-book.html`.)

[101] O. Goldreich, S. Goldwasser, and S. Halevi. Collision-Free Hashing from Lattice Problems. *ECCC*, TR95-042, 1996.

[102] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33, No. 4, pages 792–807, 1986.

[103] O. Goldreich, S. Goldwasser, and S. Micali. On the Cryptographic Applications of Random Functions. In *Crypto84*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 276–288, 1985.

[104] O. Goldreich, R. Impagliazzo, L.A. Levin, R. Venkatesan, and D. Zuckerman. Security Preserving Amplification of Hardness. In *31st IEEE Symposium on Foundations of Computer Science*, pages 318–326, 1990.

[105] O. Goldreich and A. Kahan. How to Construct Constant Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, Vol. 9, No. 2, pages 167–189, 1996. (Preliminary versions date to 1988.)

[106] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 25, No. 1, February, pages 169–192, 1996.

[107] O. Goldreich and H. Krawczyk. On Sparse Pseudorandom Ensembles. *Random Structures and Algorithms*, Vol. 3, No. 2, pages 163–174, 1992.

[108] O. Goldreich, H. Krawcyzk, and M. Luby. On the Existence of Pseudorandom Generators. *SIAM Journal on Computing*, Vol. 22, No. 6, pages 1163–1175, 1993.

[109] O. Goldreich and E. Kushilevitz. A Perfect Zero-Knowledge Proof for a Decision Problem Equivalent to Discrete Logarithm. *Journal of Cryptology*, Vol. 6, No. 2, pages 97–116, 1993.

[110] O. Goldreich and L.A. Levin. Hard-Core Predicates for Any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[111] O. Goldreich and B. Meyer. Computational Indistinguishability – Algorithms vs. Circuits. *Theoretical Computer Science*, Vol. 191, pages 215–218, 1998.

[112] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 1, pages 691–729, 1991. (Preliminary version in *27th IEEE Symposium on Foundations of Computer Science*, 1986.)

[113] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[114] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *ECCC*, TR95-050, 1995.

[115] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, Vol. 7, No. 1, pages 1–32, 1994.

[116] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. *Computational Complexity*, Vol. 8, pages 50–98, 1999.

[117] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning Polynomials with Queries: The Highly Noisy Case. To appear in *SIAM Journal on Discrete Mathematics*.

[118] O. Goldreich, A. Sahai, and S. Vadhan. Honest-Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge. In *30th ACM Symposium on the Theory of Computing*, pages 399–408, 1998.

[119] O. Goldreich and M. Sudan. Computational Indistinguishability: A Sample Hierarchy. *Journal of Computer and System Science*, Vol. 59, pages 253–269, 1999.

[120] O. Goldreich and S. Vadhan. Comparing Entropies in Statistical Zero-Knowledge with Applications to the Structure of SZK. In *14th IEEE Conference on Computational Complexity*, pages 54–73, 1999.

[121] S. Goldwasser and J. Kilian. Primality Testing Using Elliptic Curves. *Journal of the ACM*, Vol. 46, pages 450–472, 1999. (Preliminary version in *18th ACM Symposium on the Theory of Computing*, 1986.)

[122] S. Goldwasser and L.A. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *Crypto90*, Springer-Verlag Lecture Notes in Computer Science (Vol. 537), pages 77–93, 1991.

[123] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. (Preliminary version in *14th ACM Symposium on the Theory of Computing*, 1982.)

[124] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. (Preliminary version in *17th ACM Symposium on the Theory of Computing*, 1985.)

[125] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, April, pages 281–308, 1988.

[126] S. Goldwasser, S. Micali, and P. Tong. Why and How to Establish a Private Code in a Public Network. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 134–144, 1982.

[127] S. Goldwasser, S. Micali, and A.C. Yao. Strong Signature Schemes. In *15th ACM Symposium on the Theory of Computing*, pages 431–439, 1983.

[128] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Advances in Computing Research: A Research Annual*, Vol. 5 (*Randomness and Computation*, S. Micali, ed.), pages 73–90, 1989.

[129] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of a Pseudorandom Generator from Any One-Way Function. *SIAM Journal on Computing*, Vol. 28, No. 4, pages 1364–1396, 1999. (Preliminary versions by Impagliazzo et al. in *21st ACM Symposium on the Theory of Computing* (1989) and Håstad in *22nd ACM Symposium on the Theory of Computing* (1990).)

[130] J. Håstad, A. Schrift, and A. Shamir. The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits. *Journal of Computer and System Science*, Vol. 47, pages 376–404, 1993.

[131] R. Impagliazzo and M. Luby. One-Way Functions are Essential for Complexity Based Cryptography. In *30th IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[132] R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provable as Secure as Subset Sum. *Journal of Cryptology*, Vol. 9, pages 199–216, 1996.

[133] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *21st ACM Symposium on the Theory of Computing*, pages 44–61, 1989.

[134] R. Impagliazzo and A. Wigderson. P = BPP if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *29th ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[135] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. In *30th IEEE Symposium on Foundations of Computer Science*, page 248–253, 1989.

[136] R. Impagliazzo and M. Yung. Direct Z o-Knowledge Computations. In *Crypto87*, Springer-Verlag Lecture Notes in Compute Science (Vol. 293), pages 40–51, 1987.

[137] A. Juels, M. Luby, and R. Ostrovsky. Secu ty of Blind Digital Signatures. In *Crypto97*, Springer-Verlag Lecture Notes in Comp ter Science (Vol. 1294), pages 150–164, 1997.

[138] J. Justesen. A Class of Constructive Asymptotically Good Algebraic Codes. *IEEE Transactions on Information Theory*, Vol. 18, pages 652–656, 1972.

[139] N. Kahale. Eigenvalues and Expansion of Regular Graphs. *Journal of the ACM*, Vol. 42, No. 5, pages 1091–1106, 1995.

[140] J. Kahn, M. Saks, and C. Smyth. A Dual Version of Reimer's Inequality and a Proof of Rudich's Conjecture. In *15th IEEE Conference on Computational Complexity*, 2000.

[141] B.S. Kaliski. Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools. Ph.D. thesis, LCS, MIT, Cambridge, MA, 1988.

[142] J. Katz and M. Yung. Complete Characterization of Security Notions for Probabilistic Private-Key Encryption. In *32nd ACM Symposium on the Theory of Computing*, pages 245–254, 2000.

[143] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.

[144] J. Kilian and E. Petrank. An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions. *Journal of Cryptology*, Vol. 11, pages 1–27, 1998.

[145] J.C. Lagarias and A.M. Odlyzko. Solving Low-Density Subset Sum Problems. *Journal of the ACM*, Vol. 32, pages 229–246, 1985.

[146] D. Lapidot and A. Shamir. Fully Parallelized Multi-prover Protocols for NEXP-Time. *Journal of Computer and System Science*, Vol. 54, No. 2, April, pages 215–220, 1997.

[147] A. Lempel. Cryptography in Transition. *Computing Surveys*, Vol. 11, No. 4, pages 285–303, December 1979.

[148] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, Vol. 261, pages 515–534, 1982.

[149] L.A. Levin. Average Case Complete Problems. *SIAM Journal on Computing*, Vol. 15, pages 285–286, 1986.

[150] L.A. Levin. One-Way Function and Pseudorandom Generators. *Combinatorica*, Vol. 7, pages 357–363, 1987.

[151] L.A. Levin. Randomness and Non-determinism. *Journal of Symbolic Logic*, Vol. 58, No. 3, pages 1102–1103, 1993.

[152] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, 1993.

[153] J.H. van Lint. *Introduction to Coding Theory*. Graduate Texts in Mathematics (Vol. 88), Springer-Verlag, 1982.

[154] A. Lubotzky, R. Phillips, and P. Samak. Ramanujan Graphs. *Combinatorica*, Vol. 8, pages 261–277, 1988.

[155] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[156] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudo-Random Functions. *SIAM Journal on Computing*, Vol. 17, pages 373–386, 1988.

[157] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992.

[158] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1996.

[159] R.C. Merkle. Secure Communication over Insecure Channels. *CACM*, Vol. 21, No. 4, pages 294–299, 1978.

[160] R.C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[161] R.C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), pages 369–378, 1987.

[162] R.C. Merkle. A Certified Digital Signature Scheme. In *Crypto89*, Springer-Verlag Lecture

Notes in Computer Science (Vol. 435), pages 218–238, 1990.

[163] R.C. Merkle and M.E. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. *IEEE Transactions on Information Theory*, Vol. 24, pages 525–530, 1978.

[164] S. Micali, C. Rackoff, and B. Sloan. The Notion of Security for Probabilistic Cryptosystems. *SIAM Journal on Computing*, Vol. 17, pages 412–426, 1988.

[165] S. Micali and P. Rogaway. Secure Computation. In *Crypto91*, Springer-Verlag Lecture Notes in Computer Science (Vol. 576), pages 392–404, 1992.

[166] G.L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Science*, Vol. 13, pages 300–317, 1976.

[167] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[168] National Bureau of Standards. *Federal Information Processing Standards*, Publ. 46 (DES 1977).

[169] National Institute for Standards and Technology. Digital Signature Standard (DSS). *Federal Register*, Vol. 56, No. 169, August 1991.

[170] M. Naor. Bit Commitment Using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[171] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Zero-Knowledge Arguments for NP Can Be Based on General Assumptions. *Journal of Cryptology*, Vol. 11, pages 87–108, 1998.

[172] M. Naor and O. Reingold. Synthesizers and Their Application to the Parallel Construction of Pseudo-Random Functions. In *36th IEEE Symposium on Foundations of Computer Science*, pages 170–181, 1995.

[173] M. Naor and O. Reingold. On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited. *Journal of Cryptology*, Vol. 12, No. 1, pages 29–66, 1999.

[174] M. Naor and O. Reingold. From Unpredictability to Indistinguishability: A Simple Construction of Pseudorandom Functions from MACs. In *Crypto98*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1464), pages 267–282, 1998.

[175] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Application. In *21st ACM Symposium on the Theory of Computing*, pages 33–43, 1989.

[176] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *22nd ACM Symposium on the Theory of Computing*, pages 427–437, 1990.

[177] N. Nisan and D. Zuckerman. Randomness Is Linear in Space. *Journal of Computer and System Science*, Vol. 52, No. 1, pages 43–52, 1996.

[178] A.M. Odlyzko. The Future of Integer Factorization. *CryptoBytes* (RSA Laboratories), Vol. 1, No. 2, pages 5–12, 1995. (Available from `http://www.research.att.com/~amo`.)

[179] A.M. Odlyzko. Discrete Logarithms and Smooth Polynomials. In *Finite Fields: Theory, Applications and Algorithms*, G.L. Mullen and P. Shiue, eds., Contemporary Mathematics, Vol. 168, American Mathematical Society, pages 269–278, 1994. (Available from `http://www.research.att.com/~amo`.)

[180] T. Okamoto. On Relationships between Statistical Zero-Knowledge Proofs. In *28th ACM Symposium on the Theory of Computing*, pages 649–658, 1996.

[181] R. Ostrovsky and A. Wigderson. One-Way Functions Are Essential for Non-Trivial Zero-Knowledge. In *2nd Israel Symposium on Theory of Computing and Systems*, IEEE Comp. Soc. Press, pages 3–17, 1993.

[182] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *10th ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[183] B. Pfitzmann. *Digital Signature Schemes (General Framework and Fail-Stop Signatures)*. Springer-Verlag Lecture Notes in Computer Science (Vol. 1100), 1996.

[184] V. Pratt. Every Prime Has a Succinct Certificate. *SIAM Journal on Computing*, Vol. 4, pages 214–220, 1975.

[185] M.O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, Vol. 12, pages 128–138, 1980.

[186] M.O. Rabin. Digitalized Signatures. In *Foundations of Secure Computation*, R.A. DeMillo et al., eds. Academic Press, 1977.

[187] M.O. Rabin. Digitalized Signatures and Public Key Functions as Intractable as Factoring. TR-212, LCS, MIT, Cambridge, MA, 1979.

[188] M.O. Rabin. How to Exchange Secrets by Oblivious Transfer. Tech. Memo. TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[189] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1592), pages 415–413, 1999.

[190] R. Raz. A Parallel Repetition Theorem. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 763–803, 1998.

[191] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *CACM*, Vol. 21, pages 120–126, 1978.

[192] J. Rompel. One-Way Functions Are Necessary and Sufficient for Secure Signatures. In *22nd ACM Symposium on the Theory of Computing*, pages 387–394, 1990.

[193] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Achieving Chosen-Ciphertext Security. In *40th IEEE Symposium on Foundations of Computer Science*, pages 543–553, 1999.

[194] A. Sahai and S. Vadhan. A Complete Promise Problem for Statistical Zero-Knowledge. In *38th IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1997.

[195] C.P. Schnorr and H.H. Horner. Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction. In *EuroCrypt95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 921), pages 1–12, 1995.

[196] A. Shamir. How to Share a Secret. *CACM*, Vol. 22, pages 612–613, 1979.

[197] A. Shamir. A Polynomial-Time Algorithm for Breaking the Merkle-Hellman Cryptosystem. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 145–152, 1982.

[198] A. Shamir. IP = PSPACE. *Journal of the ACM*, Vol. 39, No. 4, pages 869–877, 1992.

[199] A. Shamir, R.L. Rivest, and L. Adleman. Mental Poker. Report TM-125, LCS, MIT, Cambridge, MA, 1979.

[200] C.E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, Vol. 28, pages 656–715, 1949.

[201] M. Sipser. A Complexity Theoretic Approach to Randomness. In *15th ACM Symposium on the Theory of Computing*, pages 330–335, 1983.

[202] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, Boston, MA, 1997.

[203] R. Solovay and V. Strassen. A Fast Monte-Carlo Test for Primality. *SIAM Journal on Computing*, Vol. 6, pages 84–85, 1977. (Addendum in *SIAM Journal on Computing*, Vol. 7, page 118, 1978.)

[204] M. Sudan. Decoding of Reed-Solomon Codes beyond the Error-Correction Bound. *Journal of Complexity*, Vol. 13, No. 1, pages 180–193, 1997.

[205] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information. In *28th IEEE Symposium on Foundations of Computer Science*, pages 472–482, 1987.

[206] S. Vadhan. A Study of Statistical Zero-Knowledge Proofs. Ph.D. thesis, Department of Mathematics, MIT, Cambridge, MA, 1999.

[207] A. Vardi. Algorithmic Complexity in Coding Theory and the Minimum Distance Problem.

In *29th ACM Symposium on the Theory of Computing*, pages 92–108, 1997.

[208] U.V. Vazirani and V.V. Vazirani. Efficient and Secure Pseudo-Random Number Generation. In *25th IEEE Symposium on Foundations of Computer Science*, pages 458–463, 1984.

[209] M. Wegman and L. Carter. New Hash Functions and Their Use in Authentication and Set Equality. *Journal of Computer and System Science*, Vol. 22, pages 265–279, 1981.

[210] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[211] A.C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.